NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REAL TIME IMAGE ENHANCEMENT DURING
UNDERWATER RECOVERY OPERATIONS

by

William J. Partridge

June 1989

Thesis Advisor — Charles W. Therrien
Co-Advisor — Roberto Cristi

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution Availability of Report	
2b Declassification Downgrading Schedule		Approved for public release; distribution is unlimited.	
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* 39	7a Name of Monitoring Organization Naval Postgraduate School	
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
8a Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
8c Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No	Project No / Task No / Work Unit Accession No

11 Title *(include security classification)* REAL TIME IMAGE ENHANCEMENT DURING UNDERWATER RECOVERY OPERATIONS

12 Personal Author(s) William J. Partridge

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	June 1989	64

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Image enhancement, Contrast enhancement, Underwater Recovery Operations

19 Abstract *(continue on reverse if necessary and identify by block number)*

The development of a menu-driven real-time image processing program to be used during underwater torpedo recovery operations is described. Included are an analysis of the images to be processed, a description of the hardware and software tools available to solve the problem, the methodology used to select the most effective enhancement functions, and the results of a test of the program conducted in an underwater recovery environment. Appendices include a source code listing for the program and a User's Manual which provides hardware setup instructions, a tutorial on the use of the program, and a quick reference list of the menu options available.

20 Distribution Availability of Abstract	21 Abstract Security Classification	
☒ unclassified unlimited ☐ same as report ☐ DTIC users	Unclassified	
22a Name of Responsible Individual Charles W. Therrien	22b Telephone *(include Area code)* (408) 646-3347	22c Office Symbol 62Ti

DD FORM 1473,84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

Approved for public release; distribution is unlimited.

Real Time Image Enhancement During
Underwater Recovery Operations

by

William J. Partridge
Major, United States Army
B.S., United States Military Academy, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

The development of a menu-driven real-time image processing program to be used during underwater torpedo recovery operations is described. Included are an analysis of the images to be processed, a description of the hardware and software tools available to solve the problem, the methodology used to select the most effective enhancement functions, and the results of a test of the program conducted in an underwater recovery environment. Appendices include a source code listing for the program and a User's Manual which provides hardware setup instructions, a tutorial on the use of the program, and a quick reference list of the menu options available.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The Naval Undersea Warfare Engineering Station (NUWES) conducts testing of torpedoes on various underwater ranges in the Pacific Northwest (Washington State and Canada). On occasion these torpedoes malfunction and are lost on or under the ocean floor and must be recovered. The recovery operation involves locating the torpedo, if neccessary digging it out from under the ocean bottom, and attaching a cable to it in order to pull it out. This is all done by remote control by an operator on a surface ship using video and sonar images to guide him. The sonar signal is used for general position location and then video images are used to conduct the remainder of the operation.

There are several problems which occur during the acquisition of these images. Probably the most critical of these are caused by the artificial lighting required at depths of up to 1500 feet. Uneven lighting can produce severe shadows and bright blooming on the same image frame. Bottom texture (important for finding the entry point when the torpedo has buried itself) often cannot be distinguished. Silt and debris thrown up by the digging device can totally obscure vision for up to several hours at a time.

The tools chosen to help solve these problems were an IBM AT Personal Computer[1] with a PCVISIONplus FRAMEGRABBER[2] board installed and the ITEX PCplus[3] library of image processing subroutines. This system allows one to build and run software tailored to his her particular image processing needs. The system is relatively inexpensive (when compared to more sophisticated image processing workstations), portable, and easily installed in an office or on board a recovery vessel.

The purpose of this thesis was to develop an image processing package, using the tools described above, to provide contrast and other enhancement of video images. Because the images to be processed are used as feedback in operating a remote control device, a real-time or near real-time approach to the problem was neccesitated. In a previous thesis, [Ref. 1] Roberto Ventura suggested the use of lookup table modification as a primary solution. There were also several other non-lookup table algorithms which appeared to have potential usefulness. Once developed, the package was to be tested

---

1 Trademark of IBM

2 Trademark of Imaging Technology Inc.

3 Trademark of Imaging Technology Inc.

at NUWES and delivered to them for their use. Inherent in this requirement was a "user-friendly" presentation at the computer monitor and appropriate documentation for its operation. The text of this thesis shows an analysis of the problem, describes, in general, the tools used, and outlines the development of the final package. Further, it describes the results of the testing conducted on a recovery vessel at NUWES. The appendices include instructions for the use of the package and the C language code for the program.

II. ANALYSIS OF THE PROBLEM

After observing a recovery operation, receiving comments from operators, and examining video tape of typical underwater conditions, three viewing regimes were identified. These were viewing through turbid water (due to disturbance of bottom silt), clear water viewing of the bottom while searching, and clear water viewing of shiny objects which may cause blooming. Each regime was analyzed to determine what problems, if any, could potentially be solved. The calculation and display of histograms, a primary tool in the analysis, will be discussed in a later chapter.

The most challenging regime is that of viewing objects through turbid water. Often the density of particles in the water is so great that all incident light is scattered or reflected before it can reach the area or object to be observed. In this case the video system is of no use and some other portion of the electromagnetic spectrum should be considered for image acquisition. However, if even a little light is reflected from the object of interest back to the camera there is potential for enhancement of the resulting image. The histogram of a typical turbid water image, as shown in Figure 1, provides information which could lead to the method used to acheive this improvement. The tight distribution of pixel values around the center range of intensity levels readily lends itself to contrast enhancement using look-up table modification techniques.

When searching for a torpedo that has buried itself under the bottom, the general location is first determined using sonar. Once in the vicinity, however, it is very useful if the entry point can be found. This is often difficult since the mud closes in over the top of the torpedo. It would be helpful if some method could be found to better identify marks on the bottom which could be entry points. Figure 2 shows the histogram of a typical image of the bottom. Again, most pixel values are concentrated around the center range of intensity levels and this provides excellent opportunity for enhancement by look-up table modification. Also, an enlarged view of a portion of the image could be potentially helpful in object identification. This could aid in identifying a partially uncovered object during the digging operation.

Blooming is a phenomona caused by the specular reflection of bright lights from shiny objects. It often occurs after the torpedo is found while attempting to attach a cable to it. The result is a "white out" over a significant portion of the image. The current operational solution to this problem is to dim the entire image on the monitor

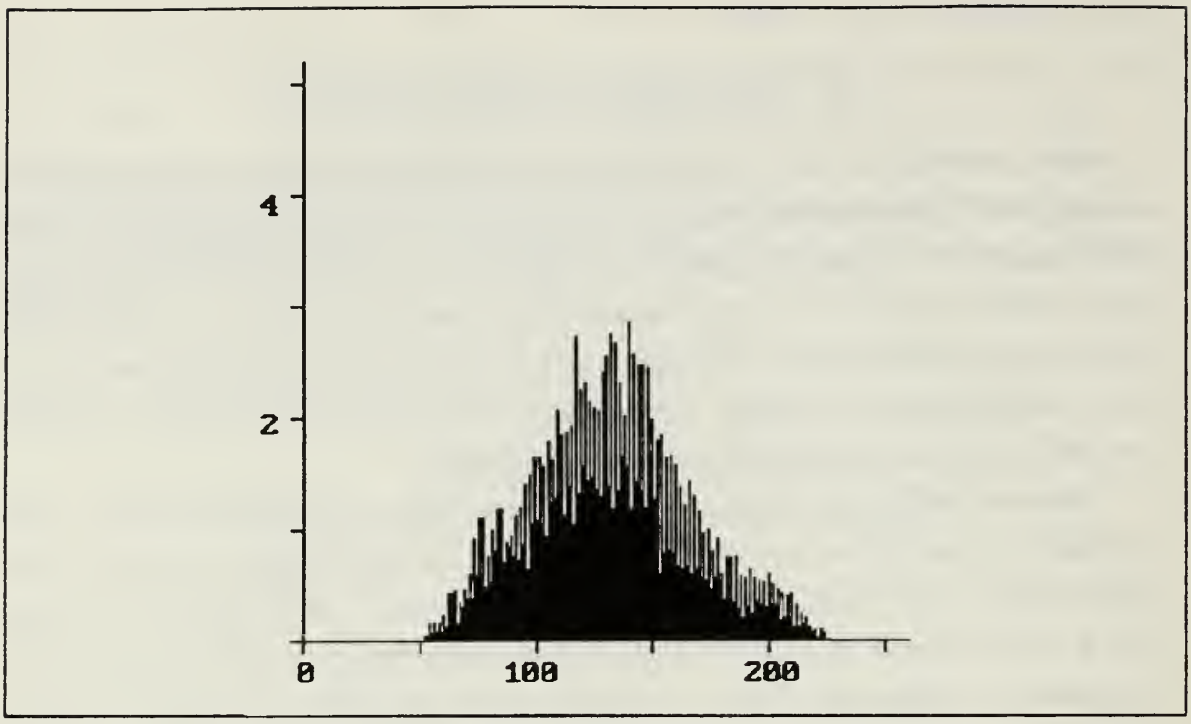

Figure 1. Histogram of a Turbid Water Image

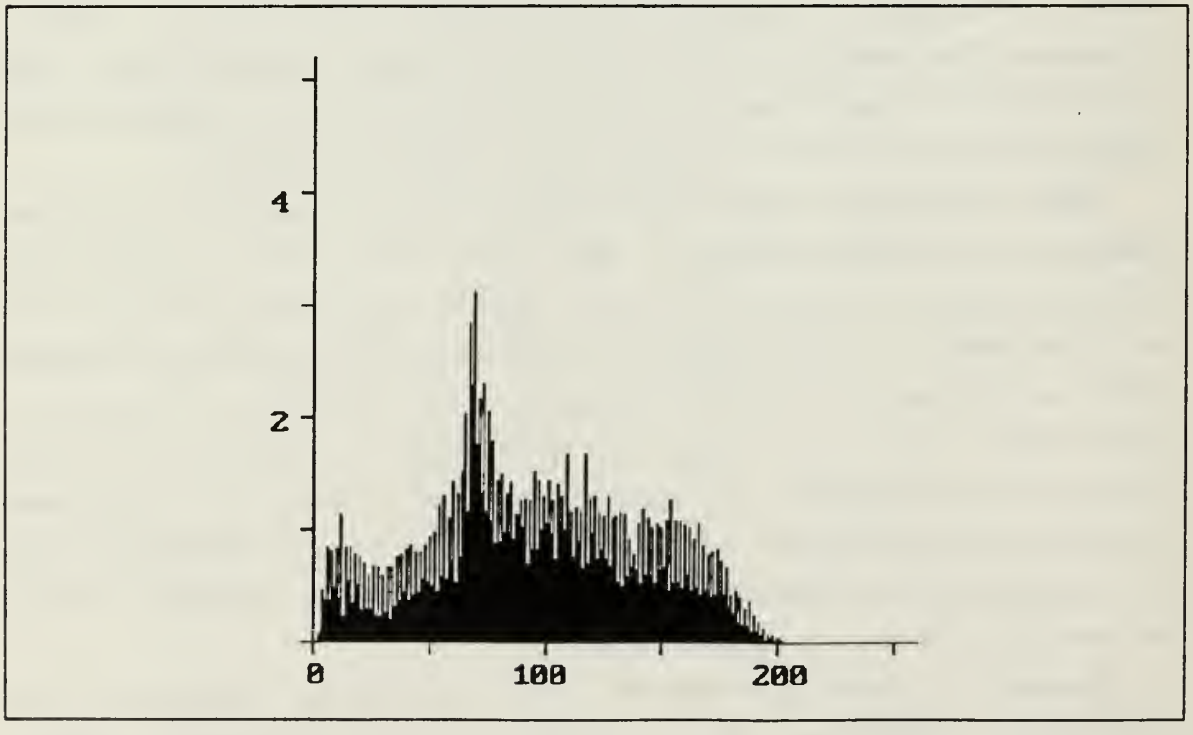

Figure 2. Histogram of a Bottom Image

using the brightness control. This reduces the detail available from the darker portions of the image. The histogram of an image with blooming, shown in Figure 3, also suggests that a look-up table modification could be performed to reduce the brightness of higher intensity pixels without dimming the lower values.

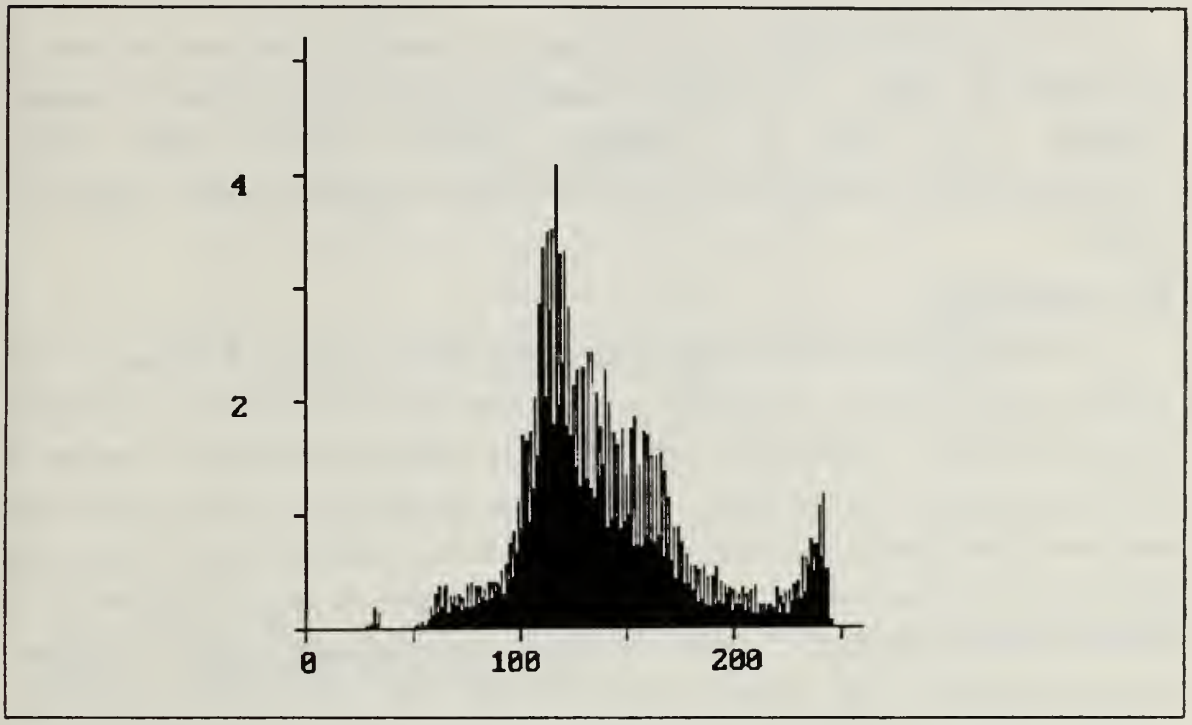

Figure 3. Histogram of Image With Blooming.

III. DESCRIPTION OF THE IMAGE PROCESSING SYSTEM

A. GENERAL

This chapter describes the tools to be used in the attempt to improve the quality of the underwater images. The image processing system, shown in Figure 4, includes hardware in the form of a personal computer with the ITEX PCplus FRAMEGRABBER installed, and a library of image processing software called ITEX PCplus.

B. HARDWARE

The framegrabber (enclosed within the dashed line in Figure 4) is the heart of the image processing system. It is installed in a host computer for control of its components and for disk access. Compatibility with the host is ensured by placement of jumpers in the proper locations on the board. This also sets the memory base and register base addresses. The framegrabber receives video input from a source (camera or video tape player) and sends video output to a monitor. At the input the image is digitized by an analog-to-digital converter. It is then sent through the input lookup table (LUT). There are eight hardware lookup tables available on the input side. One is selected for use at any given time and can be loaded with the desired values from the computer. Values may also be read from the selected lookup table and stored to disk. After passing through the input lookup table the image is placed into one of two memories, A or B, whichever is selected from the computer. The contents of these memories are accessible from the computer for analysis, processing, or storage to disk. Once an image is in a selected memory, it may be sent for display. Enroute it passes through one of eight output lookup tables which operate in the same manner as the input lookup tables. Finally it is converted to an analog image to be displayed on the video monitor. In the case of a continuous video image from the source this entire process may occur every one-thirtieth of a second, presenting a continuous image to be displayed at the output. Single frames from a source or from the computer disk may also be placed into memory and presented at the output for display. [Ref. 2]

C. SOFTWARE

The framegrabber hardware is controlled using a library of functions called ITEX PCplus. These functions may be incorporated in a program using the "C" language.

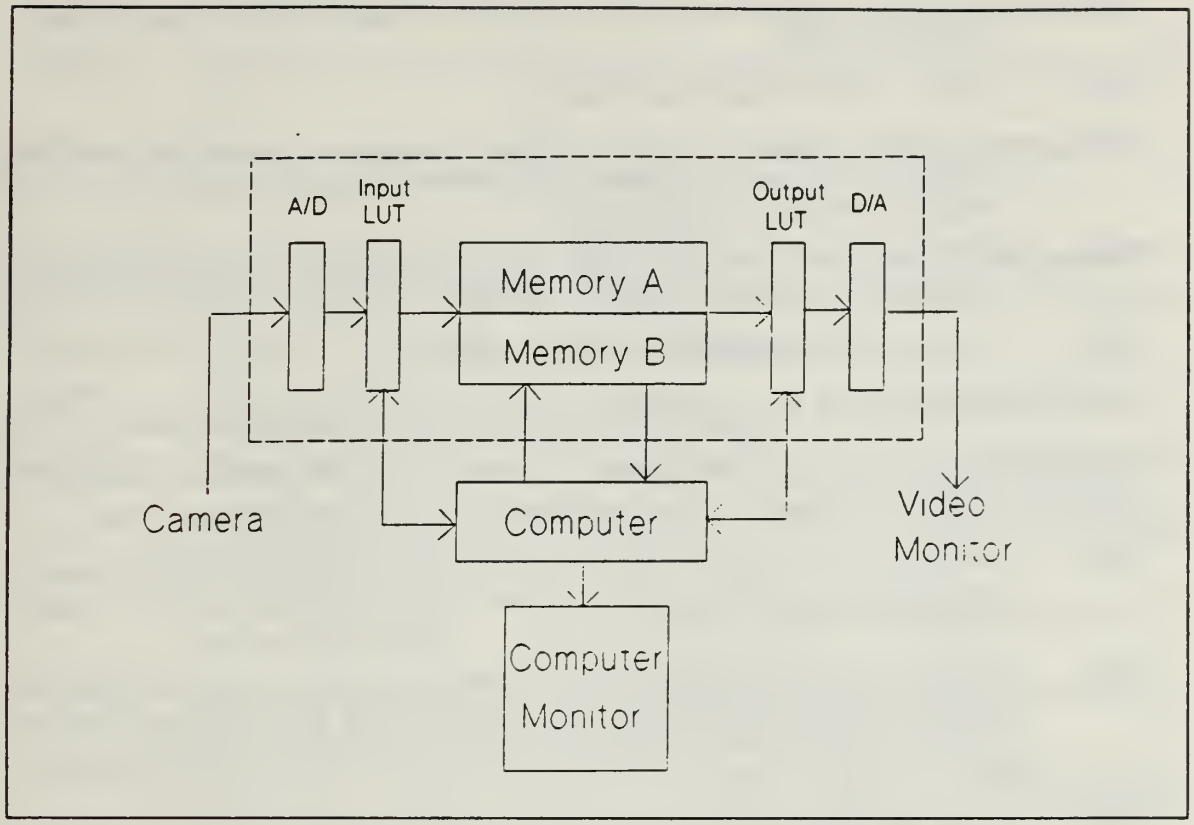

Figure 4. The Image Processing System

They perform various tasks including computer setup, image acquisition, image storage to disk, lookup table operations, and single frame image processing. Along with basic functions to place the framegrabber into operation, several others were identified as having potential for providing real time or near real time contrast enhancement. These building blocks are described here.

The setup functions configure the computer to be compatible with the framegrabber configuration, select memory size options, initialize the framegrabber, and place it into operation. They also allow selection of memory for acquisition, display, storage, and processing. A brief description of these functions follows:

sethdw Defines the hardware settings including register base address, memory base address, and memory option. Register base and memory base address arguments tell the computer how the framegrabber is configured. The memory option defines either dual 512x512 frame memories or a single 1024x512 memory.

setdim Defines the memory size. Sets memory's horizontal and vertical size and number of bits used to store each pixel.

initialize Initializes all hardware. Sets all hardware registers and lookup tables to standard values.

fgon Turns on the framegrabber memory.

select_mem Selects frame memory. Allows selection of memory for loading from disk, storage to disk, or processing. May be used to access one memory while another memory is being displayed.

display_mem Selects frame memory display. Allows selection of memory for acquisition from source and/or display to screen.

fgoff Turns off the framegrabber memory. [Ref. 3]

The image acquisition functions allow input of image data from the source. They allow continuous passage of image frames from source to video monitor, single frame storage and display, and memory clear operations. They are described as follows:

snap Acquires a single image frame from the source, stores it in frame memory, and displays it on the monitor. Allows single frame processing or storage to disk of an image from the source.

grab Acquires and displays image frames continuously up to a specified number of frames or indefinitely until new a command is given. Allows real time image processing since image frames pass continuously through lookup tables.

sclear Clears entire screen to a value. Erases frame memory and screen by placing a specified intensity level into each memory location.

saveim Saves a specified portion of an image to a file on disk. The file name is passed as an argument to the function.

readim Reads an image file from disk. The file name is passed as an argument to the function. [Ref. 3]

The lookup table functions provide the capability to select a specified lookup table for use, read lookup table values from disk, write lookup table values to disk, and modify lookup tables in various ways. These functions are described below:

setlut Selects which of eight lookup tables on the input or output bank will be active and available for storage of values.

walut Writes an array of values to a selected lookup table.

ralut Reads selected lookup table values into an array.

save_lut Saves the values in a selected lookup table to disk.

read_lut Reads lookup table values from disk and places them into the lookup table from which they were stored.

linlut Places linear lookup table values into selected lookup table.

invlut Places inverse linear lookup table values into selected lookup table.

loglut Places logarithmic lookup table values into selected lookup table.

abslut Places values into selected lookup table which invert all inputs greater than 127.

histeq Performs histogram equalization on a specified portion of an image and places computed values into selected lookup table. [Ref. 3]

Several non-real time image processing functions were also considered. These functions work on images fixed in frame memory and perform filtering, pixel replication, averaging, and histogram operations. They are described here:

sobel Performs a Sobel edge detector operation on a specified portion of an image. Displays prominent edges of the image.

sharpen Sharpens a specified portion of an image by selectively amplifying edges.

lopass Performs a low-pass filtering operation on a specified portion of an image and displays result.

zoom Enlarges by a factor of two the portion of the image in the upper left quarter of the display.

average Acquires the selected portion of a specified number of live image frames, stores them on disk, computes the average, and displays results.

histogram Computes a histogram of a specified portion of an image and places values into an array. Allows selection of spatial sampling rate in vertical and horizontal directions and size of storage bins (array).

pan Shifts the entire image horizontally by a specified amount.

scroll Shifts the entire image vertically by a specified amount.

rectangle Draws a rectangle with specified size and location on the screen and in frame memory. May be superimposed on a fixed image. [Ref. 3]

IV. DEVELOPMENT OF THE MENU PROGRAM

A. GENERAL

The concept of the "Menu" program was to place the various image processing options and the analytical tools into a format which allows option and parameter changes to be easily made and evaluated. Further, the menu format is used in the final image processing package to be delivered to NUWES.

This chapter describes how the various ITEX PCplus functions were incorporated using Microsoft[4] C 5.0 [Ref. 4] to produce the "Menu" program. It outlines how the setup functions were used to bring the board on line and perform some basic operations. Then it describes how the histogram and lookup table read/write functions were used to build the analytical tools. Finally, it shows how various lookup table modification and other image processing functions were written into the program and tested for applicability to the stated problems.

B. BASIC OPERATIONS

1. Setup

Four functions are required to define and initialize the hardware and place the board into operation. The **sethdw** function is used with arguments which define the memory base address as D0000, the register base address as 100, and select the dual memory model. It is also used in a menu option which allows the operator to interactively enter different addresses if the jumpers on the framegrabber board are configured differently. The **setdim** function is used to set the frame memory size to 512 x 512 pixel locations with 8 bits for each pixel value. The **initialize** function is included in the program to set all registers and lookup tables to standard values and **fgon** is included to turn the framegrabber memory on.

2. Image Acquisition

Several menu options were devoted to image aquisition functions. Two options use the **display_mem** function to allow the operator to choose which frame memory to display. Two other options use the **select_mem** function to allow selection of a frame memory for processing or storage to disk. Combinations of these four choices permit one image to be displayed while another is being processed. The **snap** function was used

---

[4] Trademark of Microsoft Corporation

in a menu option to allow the acquisition of a single image frame into a selected memory. Another option selects continuous acquisition and display of image frames using the **grab** function. The clear screen menu options use the **sclear** function to erase the screen and frame memory to either black or white. Finally, access to images stored on disk is accomplished with menu options using the **saveim** and **readim** functions.

C. BUILDING THE ANALYSIS TOOLS

1. Computation and Display of the Histogram

The histogram was the most important tool used in analysing the characteristics of an image. The **histogram** function provides the computed histogram values in an array, but methods needed to be devised to specify the portion of the image to include in the calculation, to display the histogram graphically, and to compare images and their histograms before and after processing.

To specify the portion of the image to be used in the histogram calculation, a rectangle whose size and position could be changed from the keyboard was superimposed on a fixed image displayed on the monitor using the **rectangle** function. This sequence (the **snap** and **rectangle** functions) was placed inside a loop to give the appearance of continuous acquisition and display. The size and location parameters of the rectangle were then used as the area arguments for the histogram function. A default option to use the entire screen was also programmed so that the rectangle routine could be bypassed if desired.

A method was also needed to display the histogram values graphically on the computer monitor at the same time the image was being displayed on the video monitor. This was accomplished in a routine which used Microsoft C 5.0 linedrawing and other graphics functions to draw vertical lines from a horizontal axis up to scaled histogram values. The result was a 256 line wide (one line for each intensity level) bar graph showing the relative density of each intensity level of the image. Examples of the histogram display are shown in Chapter 2 (Figures 1, 2, and 3).

To make a comparison of an image and its histogram before and after processing requires the use of both frame memories. An image frame is acquired into one memory using the **snap** function with a linear input lookup table. The other memory is selected (the **display_mem** function) and new values are loaded into the input lookup table using the **walut** function. Then another image frame is acquired. The time difference between the two frames is so short that the two images are virtually the same. The histogram values for each frame are then computed with the **histogram** function and

stored in two arrays. The histogram and frame memory to be displayed can then be selected from the keyboard, presenting processed and unprocessed versions of the image and histogram to be examined.

2. Manual Modification and Display of Lookup Tables

As stated earlier, the lookup table is the primary means by which the contrast of the image frames can be enhanced in real time. In order to predict and evaluate the effect of a particular set of lookup table values on an image a means of displaying those values graphically was needed. Further, it was desired to be able to change the lookup table graph interactively from the computer keyboard and evaluate the effect of these changes on the image frames.

To plot the lookup table the **ralut** function is used to read the values from the selected lookup table into an array. Microsoft C 5.0 linedrawing functions with line segment endpoints defined by array values plot a continuous line which represents the lookup table. Figure 5 shows a typical lookup table display.

Two methods were programmed to interactively modify lookup tables. The first method reverses the process used to plot the lookup table described above. Line segment end points are entered from the computer keyboard. The line is then drawn on the computer monitor. The equation for that line is computed and values are derived to be placed in the appropriate positions in the lookup table array. The **walut** function then writes the array to the lookup table. The effect on the image is immediately evident and may be evaluated. The second modification routine interactively varies the slope of the lookup table. The point where the line intercepts the top of the graph is shifted to the left or right by pressing the arrow keys on the computer keyboard. The remaining points on the line to the bottom of the graph are then shifted in the same direction, with the amount of shift varying linearly to zero from the initial shift. The shape of the line remains very close to the original, but the overall slope is changed, as shown in Figures 6 and 7. As the point by point changes are made, the lookup table array is continually updated and the values written to the selected lookup table. The effect of each change on the incoming image frames occurs immediately. The slope may also be varied in a similar manner by shifting the bottom portion of the lookup table line.

At this point the **save_lut** and **read_lut** functions were included as menu options to allow future use of particularly effective sets of lookup table derived during experimentation.

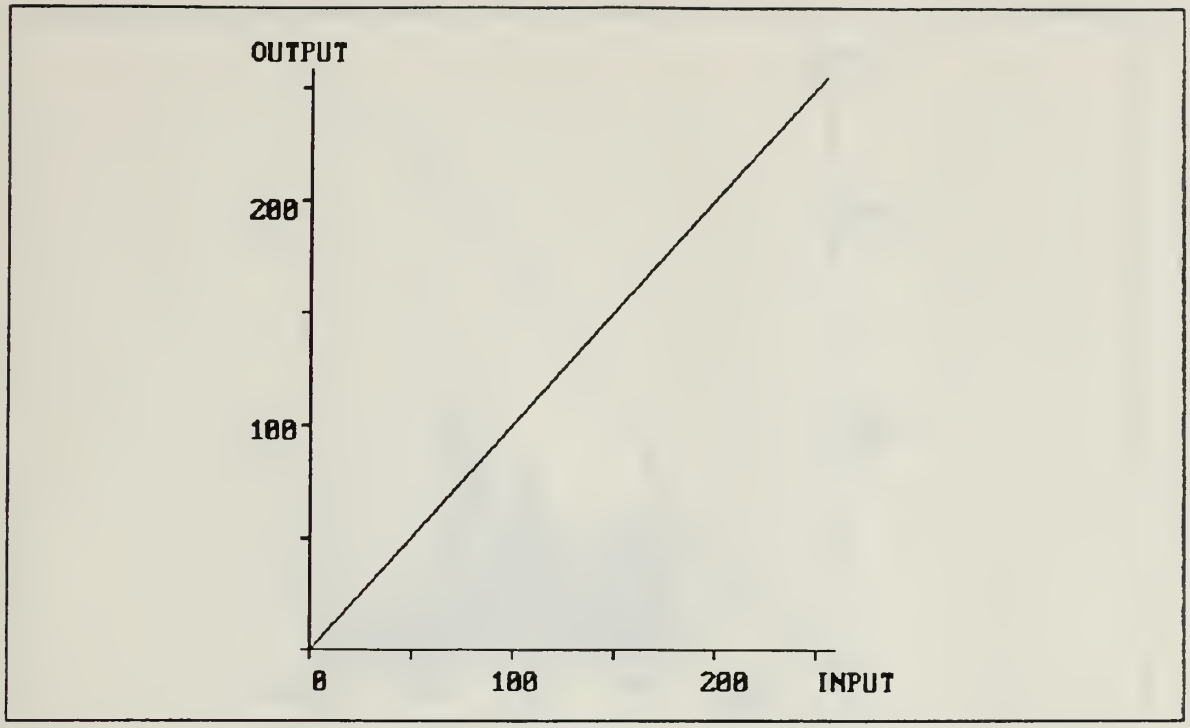

Figure 5. Typical Lookup Table Display

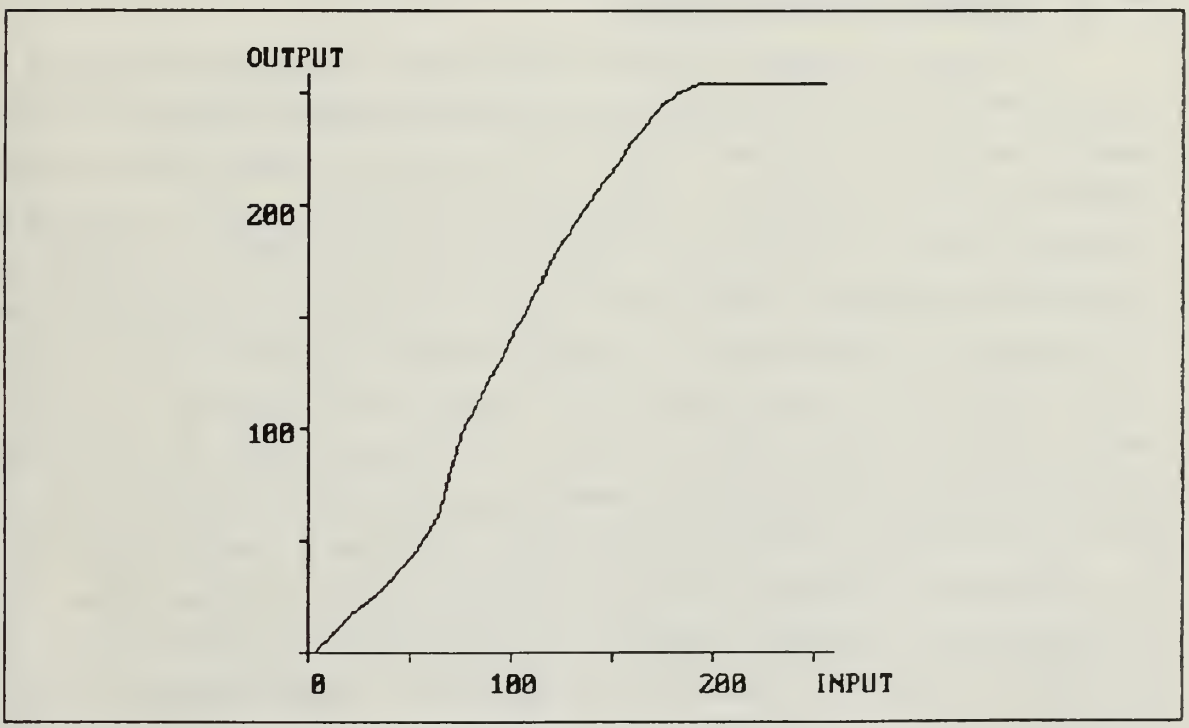

Figure 6. Lookup Table Before Slope Modification

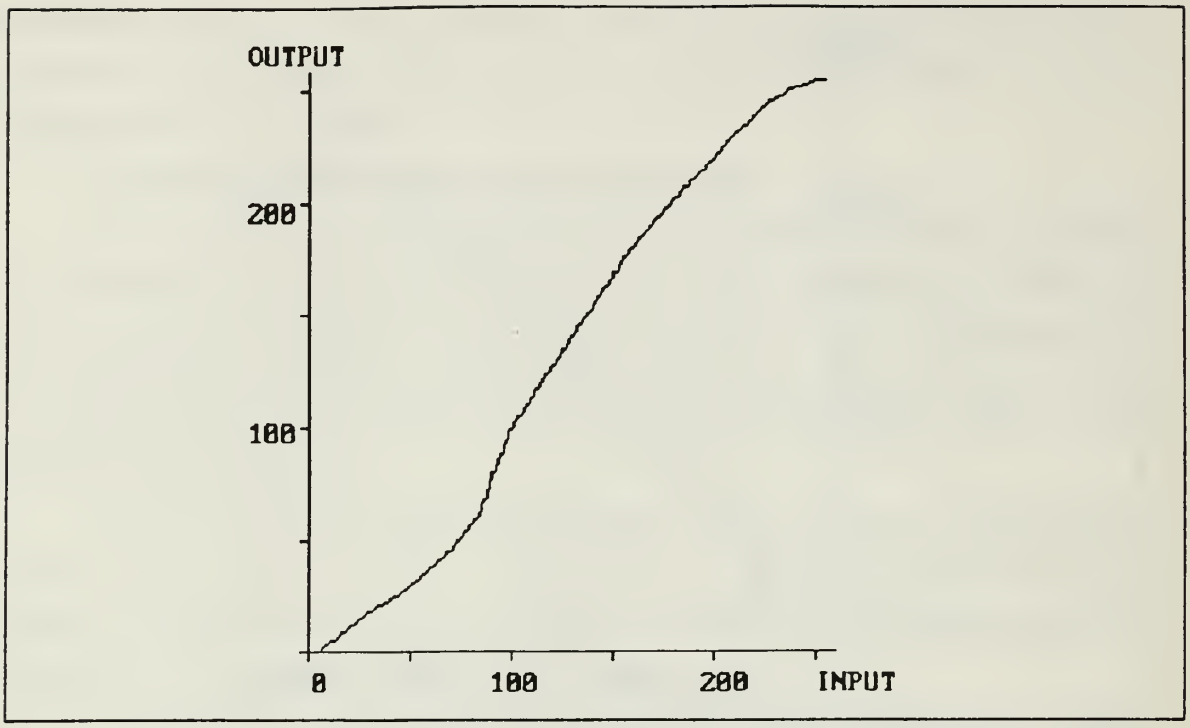

Figure 7. Lookup Table After Slope Modification

D. APPLICATION OF LOOKUP TABLE OPERATIONS

1. Lookup Table Stretch Routines

As shown in Chapter II of this thesis (Figures 1 and 2), the histograms for turbid water images and bottom images have high concentrations of pixels in the center range of intensity levels and few or zero in the outer ranges. In these situations contrast enhancement may be achieved by using lookup table stretch methods to expand the histogram to cover the full range of available intensity levels. [Ref. 5] The following experiment was performed. The histogram (shown in Figure 8) of a typical image from of a sequence of bottom images was evaluated visually for the high and low intensity values of the high pixel concentration area. Then a linear lookup table was modified using the slope varying routine described in this chapter so that the top intercept matched the high end of intensity level concentration and the bottom intercept matched the low end of intensity level concentration as shown in Figure 9. The histogram routine was then applied to compare the enhanced image with the unenhanced image. The modified histogram is shown in Figure 10. The result was an image whose contrast was significantly enhanced. The problem then became one of automating the histogram evaluation process and making it responsive to changes as they occurred in the continuously acquired images.

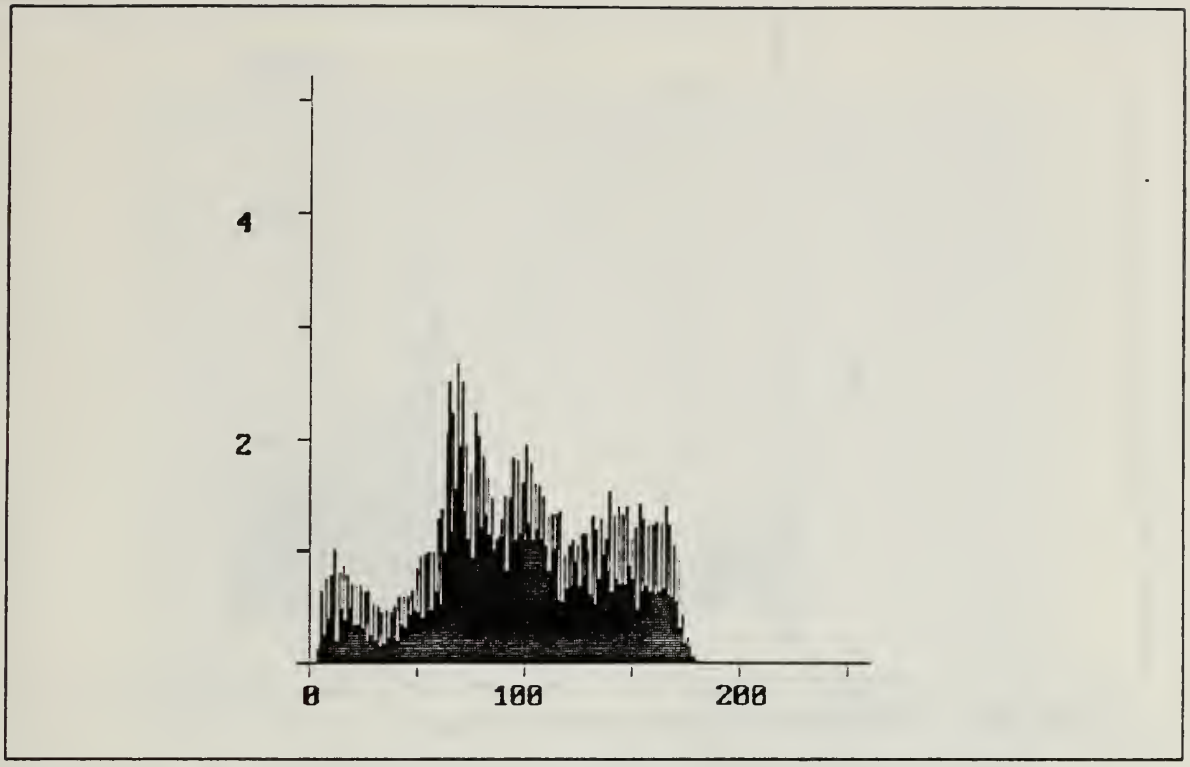

Figure 8. Histogram Before Lookup Table Stretch

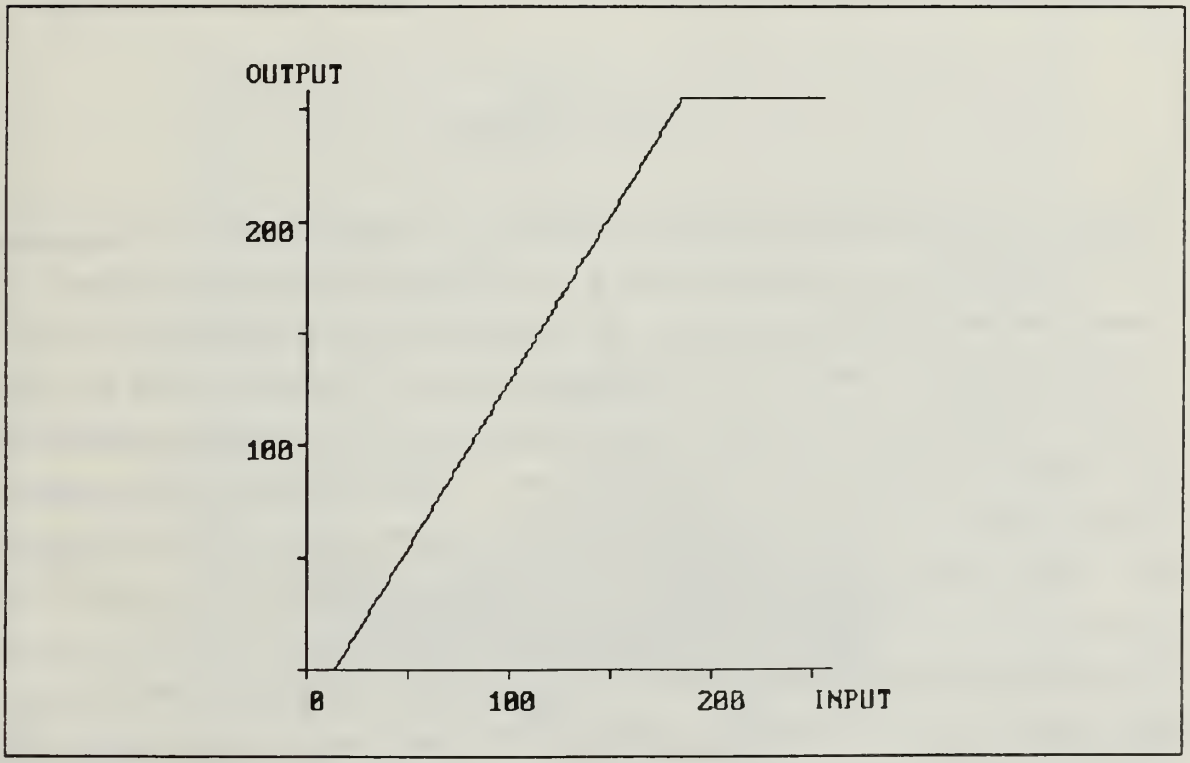

Figure 9. The Lookup Table Stretch

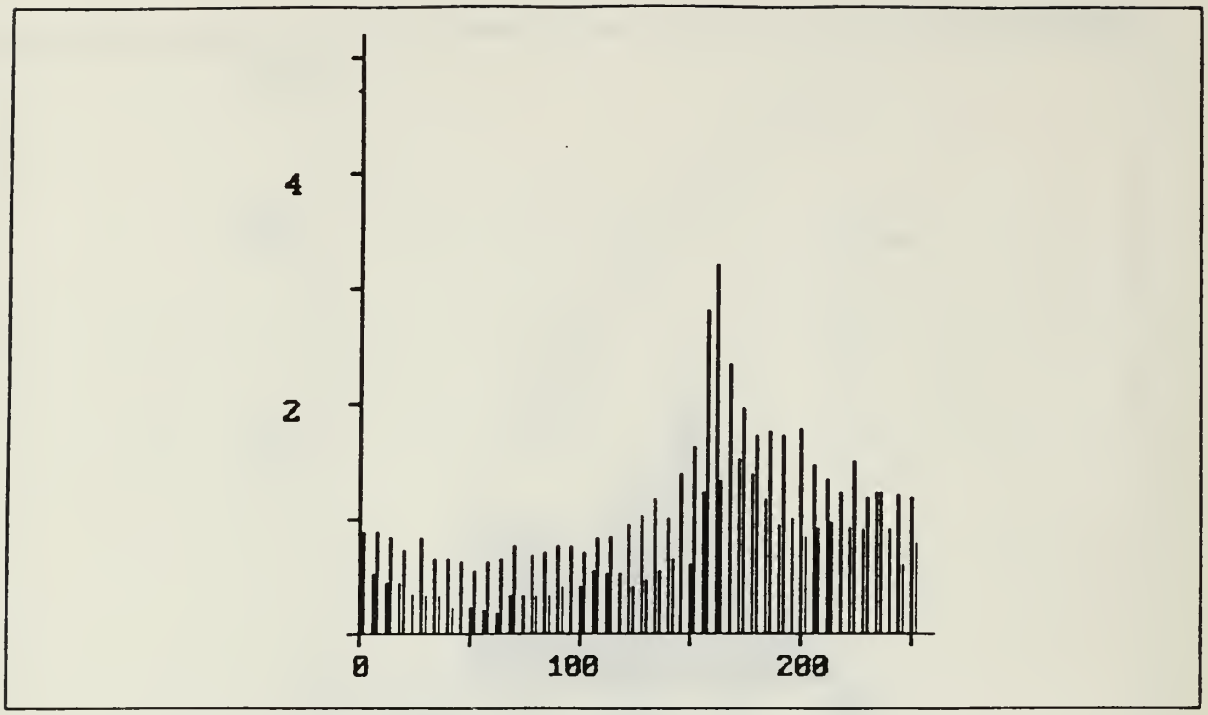

Figure 10. Histogram After Lookup Table Stretch

Histogram equalization provides the answer to the histogram evaluation portion of the problem. This process is defined by a transformation function of the form

$$V(x) = \int_0^x p(s)ds,$$

where $p(s)$ is the histogram distribution function, s is a dummy variable of integration, x is a normalized input gray level, and V is the transformed output gray level. This transformation function, when plotted for the full gray scale, gives the shape of the desired lookup table. [Ref. 6] The **histeq** function uses this concept to find the output value for each input to the lookup table by computing the normalized cumulative area up to that input intensity level on the histogram. It then loads those output values into the specified lookup table. In effect, the function steepens the slope of the lookup table plot for high concentration portions of the histogram and shallows the slope for low concentration portions, a localized lookup table stretch. Figures 11 and 12 show the lookup table after equalization and the resulting histogram. Figures 13 and 14 show an image before and after equalization. Although the overall image appears darker, details of the bottom are more readily apparent.

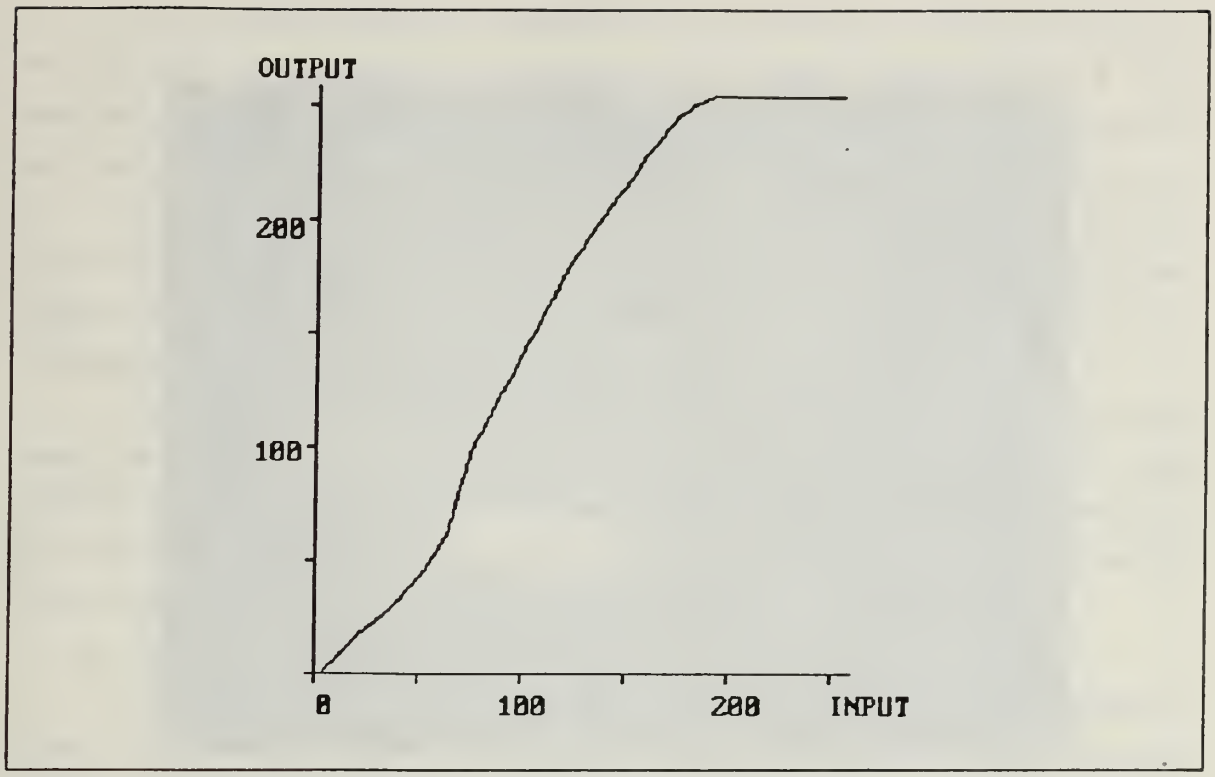

Figure 11. Lookup Table After Equalization

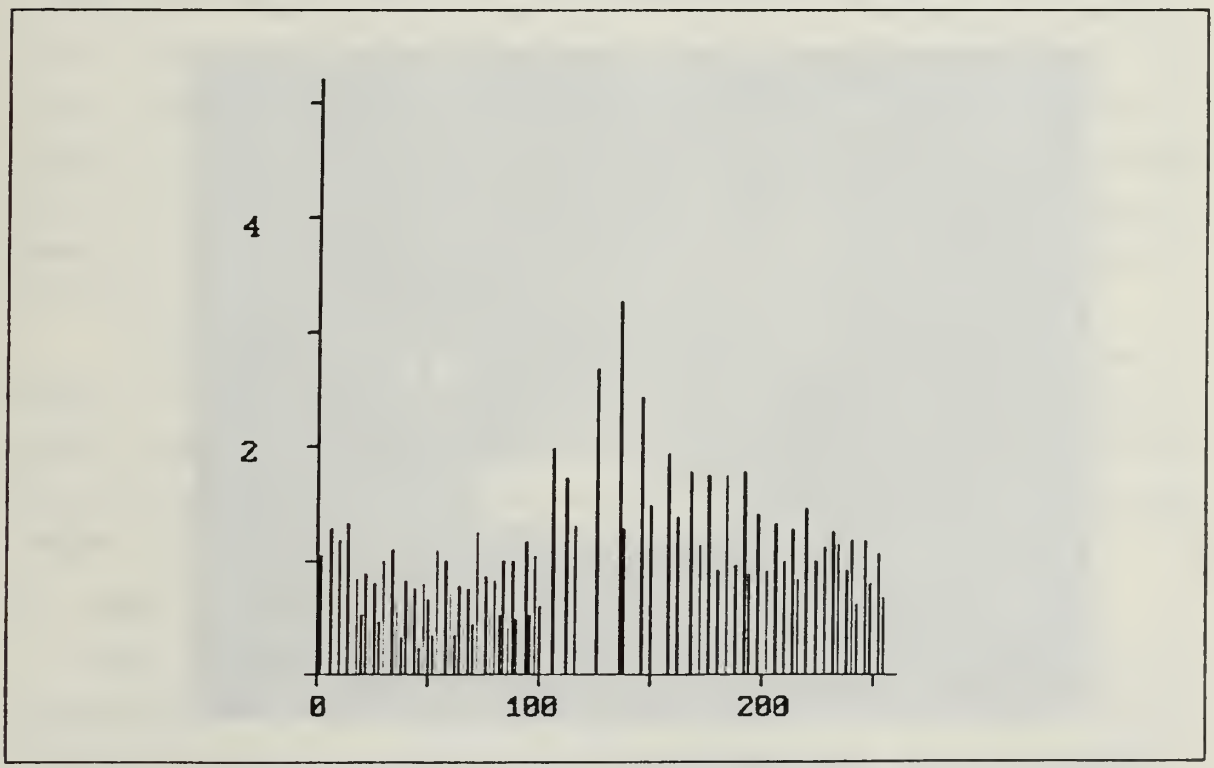

Figure 12. Histogram After Equalization

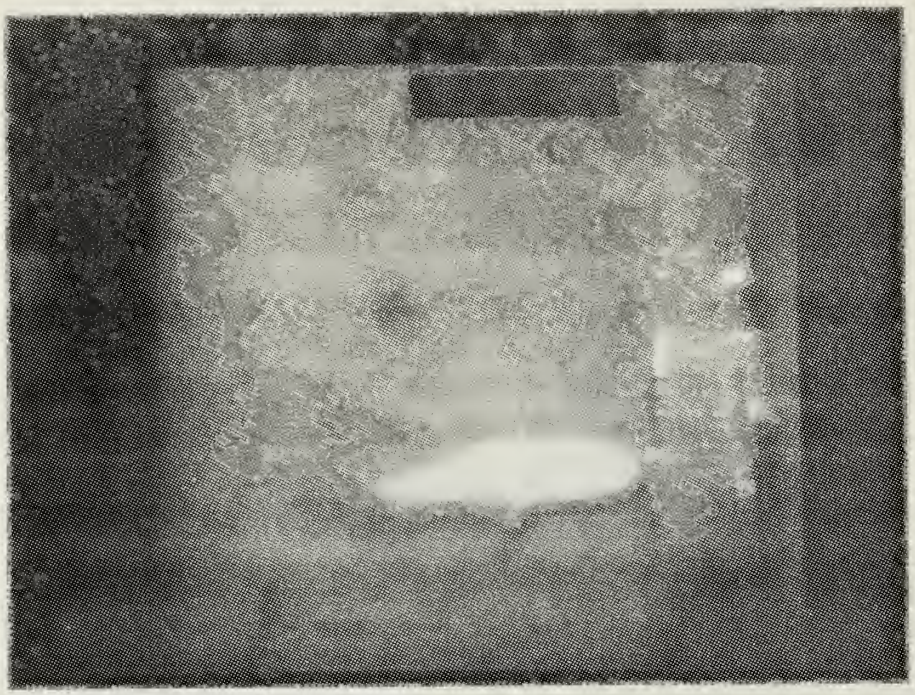

Figure 13. Image Before Equalization

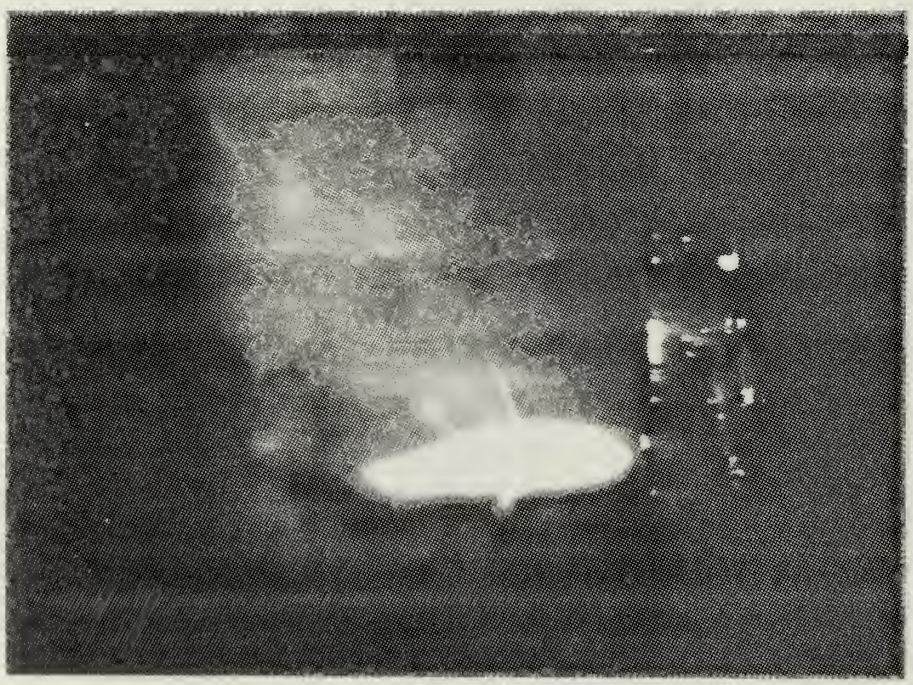

Figure 14. Image After Equalization

The problem of changing image frame content and hence changing histogram was solved by programming two alternatives for update (performance of a new histogram equalization). The first is a manual update performed from the computer keyboard by the operator as he notices the image content changing on the video monitor. The second is an automatic update where the computer routinely grabs a frame and performs a new equalization at a specified time interval. With either alternative the update occurs very quickly with less than a one second break in the continuous image acquisition as the new lookup table is loaded.

An option for equalization using the intensity levels of a specified portion of the image was also included. The user designates the portion of the image using the same rectangle described in the histogram portion of this chapter. The rectangle parameters are used to define the area used for the equalization; this results in maximum contrast for that area.

2. Other Lookup Table Routines

Chapter II of this thesis also describes the problems caused by the lighting required in the underwater environment. The histogram (shown in Figure 3) of an image in which blooming was occurring suggested that a modification to the portion of the lookup table containing the high intensity levels was in order. The lookup table drawing routine described in this chapter was used to experiment with different lookup table shapes which reduced the intensity of high value pixels. An example of one of these is shown in Figure 15. Also, the **invlut** and **abslut** functions were tried. These lookup table shapes are shown in Figures 16 and 17. The procedures most effective in reducing the blooming without affecting the remainder of the image used the inverse and absolute value lookup tables. Both were included as menu options.

Finally, the **loglut** function was tested on images which appeared dark due to shadows. The logarithmic values had a very definite brightening effect on the images, similar to that of histogram equalization on the same sequence. The **loglut** function was included as a menu option as a simple, one keystroke alternative to equalization for dark images. The logarithmic lookup table is shown in Figure 18.

E. APPLICATION OF OTHER OPERATIONS

Several non-real time image processing functions were found to be effective in enhancing turbid water and bottom images and were included as menu options. These functions all operate on a single image frame fixed in memory. The time required to perform these operations varies from four to thirty seconds depending on the operation

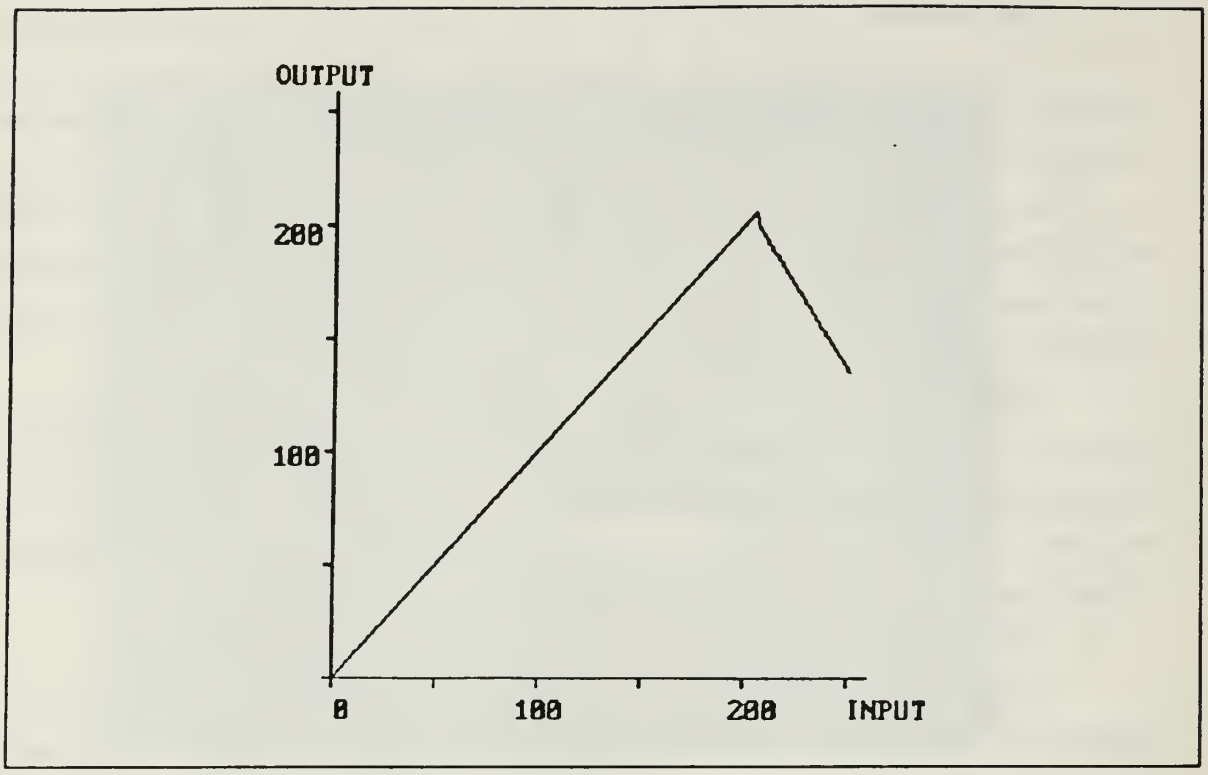

Figure 15. Experimental Lookup Table

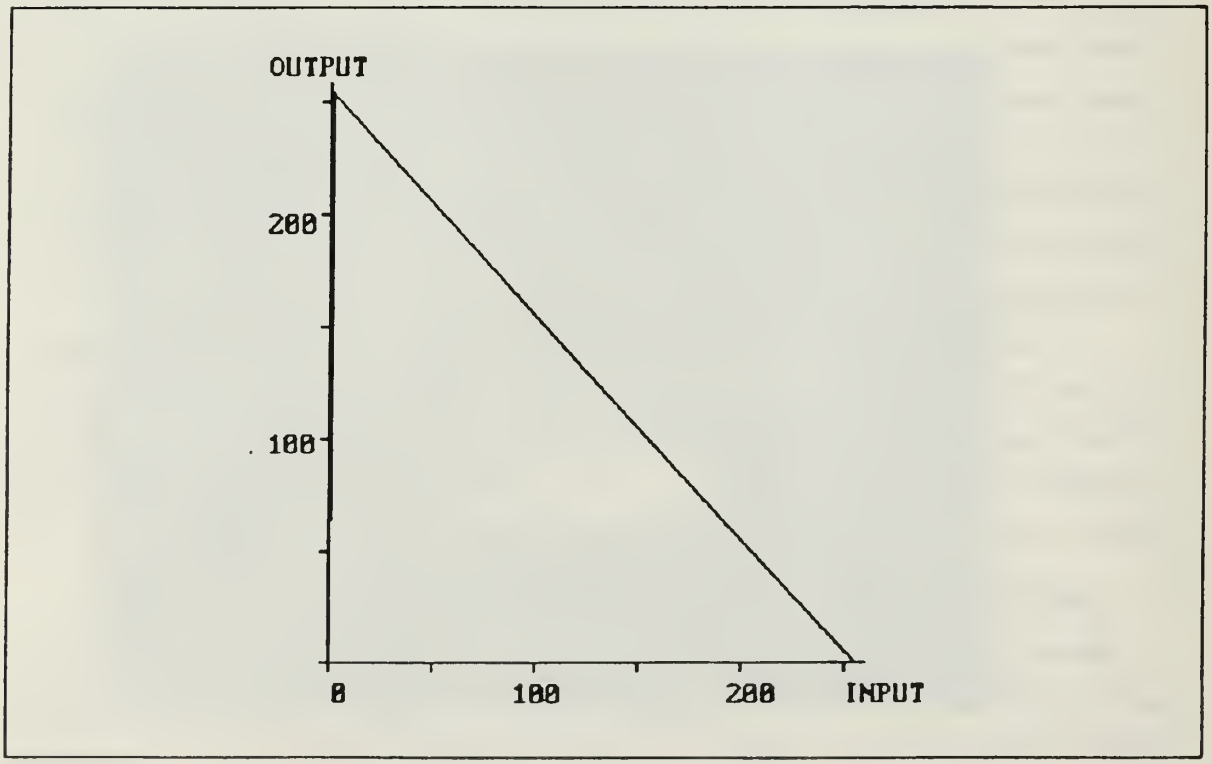

Figure 16. Inverse Lookup Table

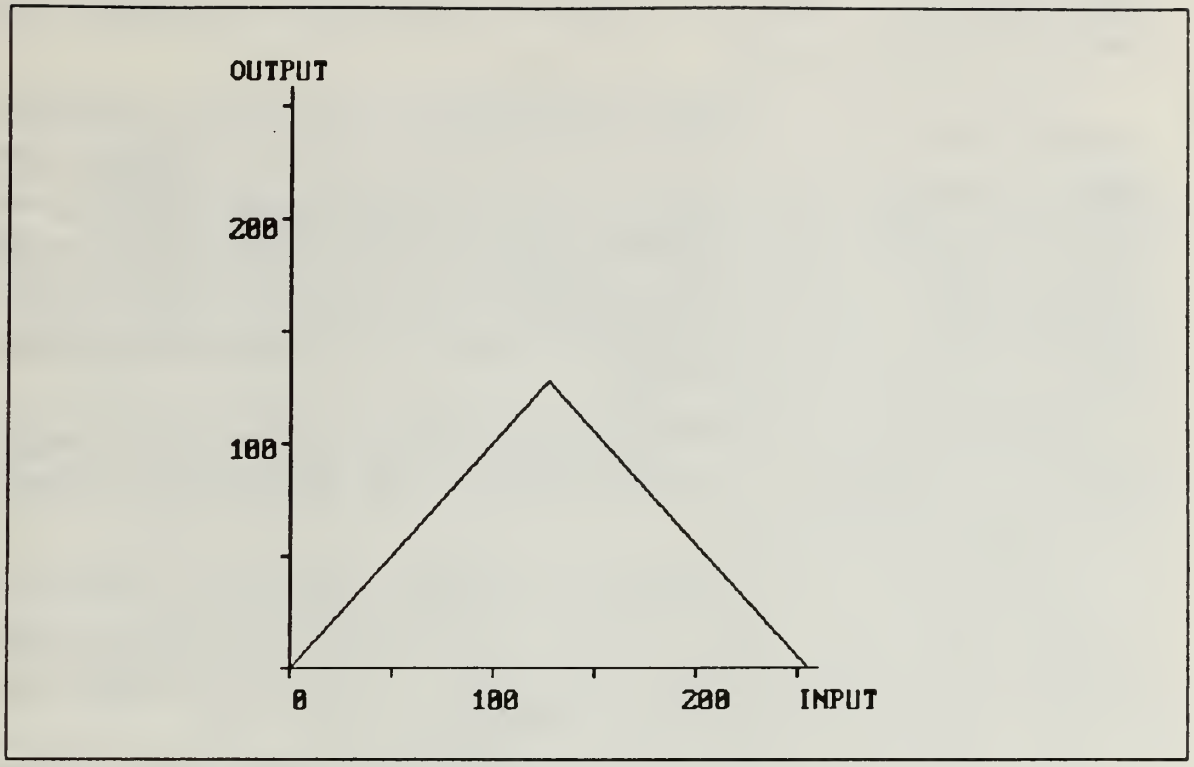

Figure 17. Absolute Value Lookup Table

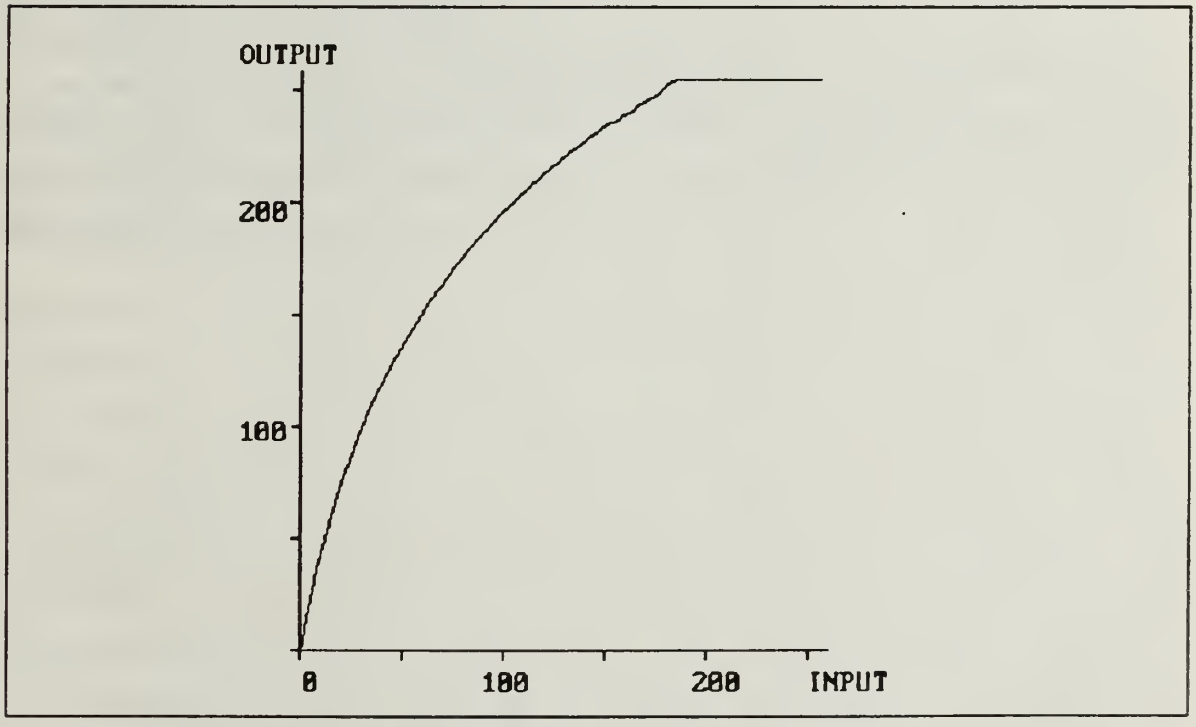

Figure 18. Logarithmic Lookup Table

and computer speed. Analysis of the effectiveness of these options was limited to visual evaluation of the actual image.

The **zoom** function was included as a menu option to assist in object identification. In order to allow selection of the portion of the screen to be enlarged, a rectangle routine similar to that used in the histogram and equalization options was programmed. In this case, however, the rectangle location parameters are sent to the **pan** and **scroll** functions to shift the image in memory to the upper left quadrant, the portion of memory used in the **zoom** function. Once the image is enlarged, it may be processed by other options that operate on a single frame fixed image.

The **sharpen** function was included as an option to provide the capability to amplify the edges of a fixed image. It is potentially helpful in object identification or in finding an entry point on the bottom.

The **lopass** function provides the capability to filter out high frequecy noise from a fixed image. It is included as a menu option as a potential tool for turbid water viewing.

The **sobel** function was included as a menu option to provide an edge detection capability. It is potentially useful in object identification.

Image averaging was included as a menu option using the **average** function. When this option is called, a specified number of image frames are acquired and stored to disk. They are then averaged, pixel by pixel, and the result is displayed on the video monitor. This option has potential usefulness when operating in turbid water with the camera mounted on a stable platform. It allows the selection of the number of image frames to be averaged up to disk memory capacity. It is the slowest of the processing options due to the high number of read write to disk operations and large number of calculations required.

V. ON SITE TESTING

A. GENERAL

An important aspect of this project was to provide the NUWES recovery section with a package that was easy to use and helpful in their specific application. For this reason a dialogue was established and maintained with the NUWES engineers and recovery section operators for the duration of the development process. The author made three visits to NUWES. The purpose of the first visit was to observe a recovery operation first hand and define the key elements of the problem. The next visit was conducted midway into the project. Its purpose was to demonstrate a prototype "Menu" program and obtain feedback about the usefulness of the various image processing options and the format of the on-screen presentation. The final visit was an on-site test of the program, conducted on a torpedo recovery vessel with cameras and equipment on the bottom under actual recovery conditions. The aft section of a torpedo was used as the test object to be viewed. The results of that test and the feedback obtained from NUWES operators and engineers are the subject of this chapter.

B. SPECIFIC APPLICATIONS

The three specific applications initially identified for processing were turbid water viewing, bottom viewing while searching, and blooming. Additionally, during the course of the development, enhancement of shadowed images and object identification became areas of interest.

The operations expected to be most effective during turbid water viewing were histogram equalization, the low pass filter, and image averaging. This portion of the test provided the most difficult environment. The histogram equalization did little to improve the the overall quality of viewing. The periods when the object was visible through the silt were so short that by the time a manual update was performed the object had disappeared. Similarly, in the automatic update mode the randomness of the short appearances of the object defeated the periodic update of the lookup table. The situation was further complicated by a horizontal dark gray synchronization bar moving through the image every two to three seconds. This reduced the level of effectiveness of the equalization since the bar's intensity values were included in the computation. In fact, the equalization process enhanced the bar. The synchronization bar had not been visible during the previous on board visit or on videotape and so may have been a

problem peculiar to the particular equipment used on this test. The low pass filter also proved to be of limited value in this environment. Again, the periods when the object was visible at all were so short that it was practically impossible to freeze an image to be processed.

Image averaging met with the highest degree of success during the turbid water portion of the test. With the recovery device (camera platform) sitting firmly on the bottom, the image average routine was able to capture those fleeting moments when the object was visible along with the time when it was not. The averaging computation then, to some extent, removed the turbidity from the image. The result was a non-real time image in which the object was recognizable. The time elapsed from the beginning of the process, averaging six frames, was approximately one minute. This was attributed to a relatively low computer speed (6 megahertz) and the read/write operations to the hard disk.[5]

The operations included in the program for enhancement of images during bottom viewing were histogram equalization and image sharpening. During the test a camera was pointed at the bottom and these operations were performed. The histogram equalization routine produced good results in increasing the contrast, making bottom features more easily identifiable. The synchronization bar again reduced effectiveness, but the value of the equalization was proven for this application. The sharpen routine was also helpful in the enhancement of bottom images, although this routine does not operate in real time. It was concluded that both of these routines would be of considerable value in identifying torpedo entry marks on the bottom.

The inverse and absolute value lookup table operations were included in the "Menu" program to help solve the problem of blooming. During the test an attempt was made to induce blooming by directing the lights towards the test object. However, due to the lighting angle and relatively dull surface of the object, severe blooming as seen on videotape did not occur. The inverse and absolute value operations performed as expected on the minimal blooming induced, and operator feedback indicated that these operations would have some utility during an actual recovery. Interest was also expressed in the capability to experiment with different lookup table shapes using the lookup table drawing routine and save those which helped solve the blooming problem.

---

[5] An attempt to speed up the image averaging operation is described in part C of this chapter.

The zoom operation was also demonstrated and although there was no specific application for its use during the test, it was agreed that it would be helpful in certain instances during recovery operations.

C. USER FEEDBACK

Upon completion of the test, comments were solicited from the recovery section operators and engineers concerning ways to improve the "Menu" program. The suggestion was made that the program's menu presentation could be revised to allow the use of a mouse to make selections and move rectangles and pointers on the video monitor. A high degree of interest was also expressed in improving the image averaging routine. Methods suggested included compensating for motion of the camera platform, using weighted averaging techniques, and speeding up the process. Once ashore, an attempt to provide more speed was made by installing the framegrabber board on a 20 megahertz Compaq 386 computer and copying the "Menu" program to a virtual disk. The combination of higher computer speed and use of RAM for temporary image storage produced a significant reduction in the time required to perform the operation (ten frames were averaged and the result was displayed in ten seconds).

Another area of interest was the potential to use the color capability available by using three framegrabber boards. Suggestions ranged from coloring the rectangle superimposed on image frames during some operations to creating pseudocolor images.

In general, there was enthusiasm expressed for the use and continued development of the capability provided by this image processing system.

VI. CONCLUSIONS

The problem of enhancement of underwater images during torpedo recovery operations was analysed and broken down into processing for three specific viewing environments. These were viewing through turbid water, viewing of bottom features, and viewing when blooming occurs. The tools available to solve these problems were then studied. These included an IBM AT Personal Computer with the PCVISIONplus FRAMEGRABBER installed and a library of image processing functions called ITEX PCplus. The "Menu" program was then developed by incorporating the image processing functions into an interactive program with a menu format. The program provides convenient and easy access to the image processing functions provided by ITEX PCplus. It also provides capability for automatic or manual equalization, a segmentation routine for improved contrast on a selected portion of an image, graphical input of lookup tables, modification of existing lookup tables, and storage of lookup tables to disk. The histogram and lookup table display routines were built and included as analysis tools. These provide on screen display of lookup tables and histograms and the capability to display enhanced and unenhanced versions of an image and their respective histograms for comparison purposes. Some non-real time functions included were image averaging, a Sobel edge detector, a low pass filter, a sharpening routine, and a zoom feature which allows interactive selection of the portion of an image to be enlarged. Documentation for the "Menu" program was written in the form of a User's Manual (see Appendix A) which includes framegrabber installation instructions, a tutorial on the use of the "Menu" program, and a quick reference list.

A test of the final package was conducted aboard a recovery vessel at NUWES with cameras and equipment on the bottom under actual recovery conditions. The program was found to be particularly effective when viewing bottom features and during blooming. The equalization procedures, lookup table drawing capability, absolute value lookup table modification, image averaging, and the zoom function were all determined to be potentially valuable tools for actual recovery operations. Appreciation was also expressed for the documentation provided with the program.

More development was called for in the turbid water viewing environment, specifically with regard to the improvement of the image averaging routine. The current function averages a finite number of image frames chosen by the user, each weighted

equally. All the frames are stored on the hard disk and then read back into RAM for averaging. The speed of the process is limited by the clock speed of the computer and the read/writes to and from disk. A stable camera platform is essential or the resulting image becomes extremely blurred. Improvements could include continuous weighted frame averaging, compensating for camera motion by cross-correlating image frames, and use of a faster computer with a virtual disk for increased speed.

The "Menu" program with documentation was delivered to NUWES for operational use.

APPENDIX A. USER'S MANUAL

A. GENERAL

The "Menu" software package is a menu-driven program which provides real time continuous video image processing using lookup table modification techniques. It also provides limited non-real time single frame enhancement capability. Control of the routines in "Menu" varies from manual (requiring operator input to make an adjustment) to automatic or "hands-off" operation. The processing routines were chosen to provide maximum effectiveness in situations where the images' pixel values are limited to a relatively small region of the total gray scale range. Typically this would include images of objects which have actual small variations in shading or those whose contrast has been reduced by obscuration or dim lighting.

B. EQUIPMENT REQUIRED

The following equipment is required to use the "Menu" program:

- IBM Personal Computer AT, XT, or PC[6] or 100% compatibles with Extended Graphics Adaptor card and monitor.
- Imaging Technology PCVISIONplus Frame Grabber[7] card with standard cable. (NOTE: The older model PCVISION Frame Grabber will not work.)
- Analog video monitor (such as SONY Trinitron[8] Model 1271Q).
- Video source (camera or video tape player).

C. SIMPLE SETUP

The PCVISIONplus Frame Grabber User's Manual provides all the information required to place the Frame Grabber into operation. Here is a much condensed version of that procedure with some helpful hints to ease the way.

1. Getting the Board Ready

Figure 2-1 on page 2-2 of the Frame Grabber Manual shows the jumper locations on the board and Table 2-1 on page 2-3 tells how they are configured from the factory. There are three position functions to be concerned with:

---

6 Trademark of IBM

7 Trademark of Imaging Technology Inc.

8 Trademark of SONY

- Since an EGA card is needed to run "Menu" the memory base address must be changed on the Frame Grabber board. Address D0000 is recommended since this is the default value in the software. To change to address D0000 simply insert a jumper at location J8. If other than D0000 is chosen it should be noted for entry each time "Menu" is run.
- "Menu" has a default register base address of 100. To change the FrameGrabber board insert a jumper at location J11. Again, if 100 is not used, the chosen value will have to be entered each time "Menu" is run.
- If an AT host is being used a jumper is required at location J20. If an XT or PC is being used no jumper should be installed at J20.

2. Installing the Board

Pages 3-1 through 3-3 of the FrameGrabber Manual provide a detailed description for installing the board. A summary is as follows:

- Turn off and unplug the computer and periphials.
- Remove the cover.
- Place the board in any open slot where it fits and install screws.
- Reinstall cover.
- Plug in and turn on computer.

3. Cable Installation

Page 3-5 of the Frame Grabber Manual shows how to connect the cable. Here's the bottom line:

- Plug the rectangular female connector on the cable into the top connector protruding from the board.
- Connect the white labeled BNC connector to the video source output.
- Connect the green labeled BNC connector to the 'line' input (not the 'green' input of the red, green, blue connections).
- Do not connect the red- or blue-labeled cables.

4. Running the Diagnostics.

Page 3-10 of the Frame Grabber Manual describes the procedure for running the diagnostic software included with the Frame Grabber. Since the factory configuration has been changed as described above, the command <configure> will have to be entered followed by the register base address (100) and the memory base address (D0000). The diagnostics take several minutes to run, so be patient. As stated in the manual, system memory or other devices mapped into the same memory space or register addresses will cause some tests to fail.

D. LET'S TRY IT OUT

This is an introduction to the "Menu" program. It lets you walk through some of the features of the program; sort of a hands-on familiarization. Turn on your source (camera or VCR) and video monitor now to let them warm up. On your computer go to the directory where you want to keep the program and copy 'menu.exe' there.

1. Load the Program

To get started simply type the word 'menu' followed by <enter>. No <enter> is needed for any selection from here on out unless otherwise noted. The menu should now be displayed on the computer monitor. If you are not using the memory base address (D0000) and register base address (100) as described in paragraph C above, hit 'K' to change setup and follow the instructions on your monitor.

2. Live Action

Menu selection 'E' provides continuous acquisition and display of the incoming real time video images. Hit 'E' and you should see live action from your camera or VCR (if you are using a tape, make sure you press the play button first).

3. Take a Snapshot

Now let's freeze the action. Hit 'F' for snap. This acquires and displays a single frame. You should see a stop action snapshot on your video monitor. The image is also stored now in framegrabber memory A, the default memory chosen when you started up.

4. Take Two Snapshots

Now hit 'D' for display memory B. Take another snapshot with the 'F' key. The second snapshot is now displayed on the monitor and stored in framegrabber memory B. Try going back to memory A (the 'C' key) to make sure the first snapshot is still there. Note that we are using the display memory keys and not the select memory keys to perform these operations. More on select memory later when we start storing images.

5. All Black

Let's erase one of our snapshots. Display memory B (the 'D' key) and then hit the 'G' key for screenclear(0). This clears the screen and framegrabber memory B to black or intensity level zero.

6. All White

While still in memory B hit the 'H' key for screenclear(255). This erases the screen and memory B to white or intensity level 255. Darker gray shades have lower intensity levels while lighter shades have higher intensity levels.

7. Put Your Snapshot in the Photo Album

Saving an image to disk is analogous to putting your snapshot in a photo album. You should still have an image in frame memory A. If not, use the display mem A and snap routines to get one. Now hit the 'A' key to select memory A. This selects memory A for data manipulations or storage. For our purposes we have selected the image data in memory A to store on disk. Now hit the 'I' key and you will be asked to enter an image name. Type a name (e.g., 'image') to store the image on the current disk and directory or path and name (e.g., 'a:image') to store it in a different place. Then you'll be prompted for a comment. You may give it a label or simply type 'none' (note that <enter> is required for each of these inputs). The disk drive selected should make some noise and your snapshot is safely tucked away in the "photo album" (i.e., the image file is written to the disk).

8. Open the Album to See Your Snapshot

Let's see if the image was really saved. First erase both frame memories as described in paragraph 5 or 6 above. Then select a memory using 'A' or 'B'. Now hit the 'J' key. Type the image file name (followed by <enter>) to read the image from disk. The image should appear on the video monitor. If it does not, display the memory you selected (the 'C' or 'D' key) and the image will appear.

9. Simple Processing

Now let's process real time video images using some simple lookup table modification techniques. Hit the 'L' key - its called analysis, but really it just lets you see how your image is being changed. The graph you see on your computer monitor is the current lookup table. The horizontal coordinate represents input pixel values and the vertical represents output values.

On the top of the computer monitor screen are the current options available for selection. The snap ('F') and grab ('G') options work the same as on the previous menu. Try them! Now try the invlut ('I') option and see how the lookup table changes (the blue line is the current lookup table and the green is a reference linear lookup table line). The effect on the image is just like looking at the negatives of a photograph, except that we still have live action. Now hit the 'L' key for linear lookup table. We are back to unprocessed live images.

Keys 'P' (logarithmic) and 'A' (absolute value) also modify the lookup table. You can experiment with these functions to see the effect on the real time image and their corresponding lookup table shape.

10. Histogram Equalization

Now for the most powerful of our lookup table modification routines. Hit 'E' for equalize. You now are given two choices, default and rectangle. This determines which portion of the image will be used to do the processing. Most of the time you will probably use the default mode but let's try rectangle now just for fun. Hit 'R' for rectangle. On the video monitor you will see the live images being displayed with a rectangle overlaid. Now hit 'Num Lock' on your keyboard. Using the '2', '4', '6', and '8' keys on the number pad (the ones with the arrows) you can move the rectangle around on the screen. The corner numbers ('1', '3', '7', and'9') are used to change the size of the rectangle. If your image had a large white or black area in which you were not interested, it would lessen the impact of the equalization routine on the area of interest. So you would frame the area of interest with the rectangle. After you have placed the rectangle where you want it, hit 'S' for stop.

Now you have to choose between manual or automatic update. Try manual update first. Hit the 'M' key. The image will equalize, the new lookup table is plotted, and live action continues using the new lookup table. In this mode, as you notice the contents of the image changing you can hit the 'U' key and equalization will occur again.

Let's go back now and try the default rectangle parameters. Hit 'C' to change parameters. Now hit 'D' for default and 'C' for continuous update. You are prompted for time between updates in seconds. Enter any number 2 through 9. The computer will perform a new equalization at the interval you specify. This is intended for situations where the contents of the image are changing often and you don't want to have to remember to update. When you have seen enough updates hit 'S' for stop.

11. Before and After

Let's look at some histograms and see whats happening when we equalize. Hit the 'H' key for histogram. Again, you may choose between default or rectangle to specify a portion of the image. Hit 'D' for default. Now hit the 'L' key for computation of linear lookup table histogram. After the "COMPUTING HISTOGRAM" indicator goes away, hit 'M' for computation of modified histogram. When the indicator goes away this time, hit 'D' to display the linear lookup table histogram. This is the information used to perform the equalization. The image on the video monitor is the unprocessed version. Now hit 'E' to display the modified histogram. The video image is enhanced and the "spread out" histogram is displayed. As you toggle back and forth between 'D' and 'E', you can see graphically how equalization lets us use the full range

of intensity levels and the corresponding enhancement of contrast on the image. Convinced? Hit the 'S' key and let's try something else.

12. Modifying the Lookup Table

Hit the 'N' key. Now we'll go again to the number pad with 'Num Lock' on. Using the corner keys ('1', '3', '7', and '9') we'll change the slope of the lookup table by moving the top or bottom of the curve. To shallow out the lookup table use the '9' key to move the top to the right and the '1' key to move the bottom to the left. Notice that enhancement to the live images changes as you modify the curve. When you are finished hit 'S' to stop and the resulting lookup table is shown. If you were to develop a "favorite" lookup table which you think you might want to use again, there is a way to save it in the computer. We'll talk about that later.

13. Draw Your Own Lookup Table

Now hit 'M' to perform another form of manual lookup table modification. We'll do this by drawing lines. Using the number pad with 'Num Lock' on move the pointer on the computer monitor by hitting the '2', '4', '6', and '8' keys. This will move it one pixel in the direction of the arrows on the keys. The next key clockwise from each of these will move it in the same direction ten pixels . (E.g., The '1' key moves the pointer in the same direction as the '2' key only faster.) When the pointer is at a location where you want the line to begin hit 'B' for begin line. Then move the pointer to a location where you want the line to end and hit 'E' for end line. Now hit 'D' to draw the line and your change will take effect on your image. You may draw other lines to connect segments and build a whole new lookup table. To get rid of the clutter just hit 'S' and then reenter the routine by hitting 'M'.

Now hit 'P' to put a pointer on the live video image. To move the pointer use the number pad keys in the same manner as you did to move the pointer on the computer monitor described above. Notice on your computer monitor that the position of the pointer and the intensity level are displayed. One possible use of this capability would be to find the intensity level of an object in the image and use the draw lookup table routine to change all pixels with that intensity to another level. Again, if you were to draw a really good all-purpose lookup table, you could save it with a routine which we'll describe later. Hit 'S' to exit the video monitor pointer routine and 'S' again to exit the modify routine.

Finally, let's go back to the main menu to see the lookup table storage/selection routines and try some single frame (frozen) processing. Type 'S' to exit the analysis function.

14. Storing and Retrieving Your Favorite Lookup Tables

Remember that lookup table you designed using the modify routine a while ago? Here's how you can save it for future use. Hit the 'M' key. Type in a name (e.g. 'mylut' or 'a:mylut'), hit enter, and the values are saved. To get it back hit the 'N' key to read the lookup table values from disk. Again, type in the file name followed by enter and your lookup table will take effect.

15. Piggyback Lookup Tables

The framegrabber board has two sets of lookup tables, one as the image is put into frame memory and one as the image is output to the monitor. Up to this point we have only used the output lookup table. Now we'll modify both sets to achieve a different result. Hit the 'P' key to select the input lookup table. Then go back to the analysis routine ('L') and do a histogram equalization. Everything still looks the same as the last time we equalized, including the video image, but this time the computed values went into the input lookup table. Now exit the equalization and analysis routines to get back to the main menu. Hit the 'O' key to select the output lookup table. Then go back into analysis. You may now use any of the lookup table modification routines (e.g. invlut, abslut, etc.) and you will get a "piggybacked" lookup table effect on the image. To get back to normal, hit 'L' to linearize the output lookup table. Go back to the main menu and select and linearize the input lookup table. Note that when you use this procedure, always put values derived by equalization into the input lookup table or the equalization will be based on already modified data.

16. Zoom

Now hit the 'R' key to start the zoom function. On the video monitor you can see the rectangle we saw before. Using the number pad with 'Num Lock' on you can move the rectangle with the '2', '4', '6', and '8' keys in the direction of the arrows on the keys. The size of this one is fixed at approximately one quarter of the screen, since we will be expanding the image to four times the original. Move the rectangle to the portion of the image you wish to zoom in on and hit 'S'. The expanded image is displayed. Notice that some resolution is lost from the original because we are "zooming" digitally as opposed to changing the focal length of the camera lens. Hit 'M' to get back to the menu. Now you can process the "zoomed" image using the fixed image processing routines described below.

17. Unzoom

To return to a normal image after using the zoom routine and performing any fixed image processing, hit the 'R' key. It is essential that this be done when you are finished with a "zoomed" image or the zoom parameters will interfere with other routines.

18. Sharpen

Snap an image by hitting the 'F' key or by using the zoom routine (the 'R' key). Then hit 'S' to sharpen it. If you watch closely you can see the sharpening occurring on the displayed image.

19. Low Pass Filter

Snap or zoom another image and then hit 'T' to call the low pass filter routine. This function is designed to remove high frequency noise from a fixed image. Again, looking closely you can see the change occurring.

20. An Edge Detector

One more time, snap or zoom an image. This time hit the 'U' key to perform a Sobel Edge Filter. Note the interesting display of the image's edges on the video monitor.

21. Image Averaging

Hit the 'V' key. Then enter the number of image frames you want to average together. This number may be limited by the amount of disk space you have available (2 is a good number to start with). When the average image is displayed, you should notice that objects which passed across the field of view during the operation have faded away. The more frames you average, the less visible these objects will become. Image averaging is very effective at eliminating transient noise, but requires a stable platform for the source.

To exit the program hit the 'W' key.

E. QUICK REFERENCE LIST

This list presents the program's options in the order they appear with a brief description of their output.

1. **A for select mem A**

   Selects framegrabber memory A for storing or reading an image to/from disk.

2. **B for select mem B**

   Selects framegrabber memory B for storing or reading an image to/from disk.

3. **C for display mem A**

Selects framegrabber memory A for acquisition of image from camera and displays memory A on the monitor.

4. **D for display mem B**

Selects framegrabber memory B for acquisition of image from camera and displays memory B on the monitor.

5. **E for grab**

Continuously acquires image frames from source and displays on video monitor. Uses framegrabber memory designated by 'display mem' function.

6. **F for snap**

Acquires a single frame from source and displays it on the video monitor. Uses framegrabber memory designated by 'display mem' function.

7. **G for sclear(0)**

   Clears screen and memory to intensity level 0.

8. **H for sclear(255)**

   Clears screen and memory to intensity level 255.

9. **I to store image**

Stores contents of selected memory to disk. Prompts input of filename (including path if desired) and comment.

10. **J to read image**

Reads image from disk file to selected framegrabber memory. Prompts input of filename (including path if desired).

11. **K to change setup**

Allows change of software memory base address and register base address. On screen instructions and example inputs provided.

12. **L for analysis**

Displays lookup tables and histograms. Performs equalization and lookup table modification routines. Options as follows:

a. m = modlut

Modifies lookup table by drawing lines. Places pointer on video monitor. Move pointer to desired start point and end point using 'Num Lock' on and number pad

keys '2', '4', '6', and '8' for down, left, up, and right respectively. Use keys '1', '7', '9', and'3' for faster movement down, left, up, and right respectively.

(1) b = beginline. Marks beginning of line.

(2) e = endline. Marks end of line.

(3) d = drawline. Draws line and enters new values into lookup table.

(4) p = pointer. Places pointer on video monitor. Gives pointer location and pixel intensity level of that location on computer monitor. Use same method to move pointer as for line drawing routine above.

(5) s = stop. Exit lookup table modify routine.

b. n = mod2 lut

Changes slope of existing lookup table and implements results. With 'Num Lock' on, use number pad keys '1' and '3' to move bottom of curve left and right respectively. Use '7' and '9' keys to move top of curve left and right respectively. Hit 's' to exit this routine.

c. h = histogram

Computes and displays histograms for images processed with linear and modified lookup tables. Snaps images before and after processing and allows for comparison.

(1) d = default. Selects default portion of image for histogram computation (approximately 90% of image).

(2) r = rectangle. Displays rectangle superimposed on image on video monitor. Change portion of image used for histogram computation by altering rectangle size and location. With 'Num Lock' on, use number pad keys '2', '4', '6', and '8' to move rectangle down, left, right, and up respectively. Use '1' and '3' keys to increase and decrease height. Use '7' and '9' keys to increase and decrease width. Hit 's' to exit rectangle routine.

(3) l = com linhist. Computes histogram of image processed by linear LUT. Displays "COMPUTING HISTOGRAM" message until done.

(4) m = com modhist. Computes histogram of image processed by modified LUT. Displays "COMPUTING HISTOGRAM" message until done.

(5) d = dis linhist. Displays histogram computed above using image processed by linear lookup table. Unprocessed image shown on video monitor.

(6) e = dis modhist. Displays histogram computed above using image processed by modified lookup table. Processed image shown on video monitor.

(7) s = stop. Exit histogram routine.

d. *e = equalize*

Performs histogram equalization with update on command or automatically at specified intervals. Displays resulting lookup table after each update.

(1) *d = default.* Selects default portion of image to use for equalization (approximately 90% of image).

(2) *r = rectangle.* Allows use of rectangle to specify portion of image to be used as described in histogram rectangle routine above (paragraph 12.c.(2)).

(3) *c = continuous update.* Selects automatic update of lookup table at specified intervals. Prompts for desired interval. Hit 's' to exit equalization routine from this mode.

(4) *m = manual update.* Selects lookup table update on command.

(a) u = update— Performs new equalization and updates lookup table.

(b) c = changeparam— Returns to default/rectangle portion of image options for changing parameters.

(c) f = freezeframe— Acquires and displays a single frame (snap).

(d) g = grab— Continuous acquisition and display of image frames.

(e) s = stop— Exit equalization routine from this mode.

e. *l = linlut*

Sets lookup table to linear values. Results in unprocessed image.

f. *i = invlut*

Sets lookup table to inverse values. Produces a negative image.

g. *p = loglut*

Sets lookup table to logarithmic values. Increases image brightness.

h. *a = abslut*

Inverses intensity levels greater than 127. Reduces image brightness.

i. *g = grab*

Continuous aquisition and display of image frames.

j. *f = freeze*

Aquisition and display of a single frame (snap).

k. *s = stop*

Exit analysis routine.

13. M to store lookup table

Stores contents of selected lookup table to disk. Prompts input of filename (including path if desired).

14. N to read lookup table

Reads lookup table values from disk to lookup table they were stored from. Prompts input of filename (including path if desired).

15. O to select output lookup table

Selects the output lookup table to perform modifications on and for display in the analysis routine. When used in conjunction with 'P' (select input lookup table) allows for processing through two modified lookup tables.

16. P to select input lookup table

Selects the input lookup table to perform modifications on and for display in the analysis routine. When used in conjunction with 'O' (select output lookup table) allows for processing through two modified lookup tables.

17. Q for zoom

Enlarges chosen area of image. Provides moveable rectangle to select area for zoom. With "Num Lock" on use number pad keys '2', '4', '6', and '8' to move rectangle down, left, right, and up respectively. Hit 'z' to stop rectangle and fix/zoom image. Hit 'm' to return to main menu.

18. R for unzoom

Returns image to normal state.

19. S for sharpen

Sharpens fixed image using a high pass filter. May be used on zoomed image.

20. T for low pass

Performs a low pass filter operation on a fixed image. Reduces high frequency noise. Blurs the image. May be used on zoomed image.

21. U for edge detector

Performs Sobel edge detector operation on a fixed image. Displays detected edges. May be used on zoomed image.

22. V for image average

Averages specified number of image frames and displays results. Prompts input of number of frames to average.

23. W for framegrabber off and exit program

Terminates program.

APPENDIX B. PROGRAM LISTING

```
/*This program provides image processing options in a menu format.  It
was written to be compiled using Microsoft C 5.0.  It also uses include
files and the library of functions from the ITEX PCplus Image Processing
Package.  The Imaging Technology PCplus Framegrabber must be installed
to run the program.  Graphics in the program require an EGA card and
monitor.*/

/*List include files*/
#include <itexpfg.h>  /*from ITEX PC plus*/
#include <stdtyp.h>   /*from ITEX PCplus*/
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <time.h>
#include <conio.h>
#include <graph.h>

/*Declare variables*/
int
i,err,n,m,t[10],s[10],j,highcut,r,array[256],pp,qq,intens,c,a,b,da,db,
e,color,x1,x2,x3,x4,yy1,y2,y3,y4,mmm,nnn,res,l,low,high,blx,bly,elx,ely,
pp1,qq1,d,lowcut,start,range,intensity,y,g,h,array2[256],vt,num;
unsigned regbase;
unsigned int grp;
unsigned long membase;
char name[20],comment[200],lutname[20],lutname2[20];
short grmode;
long histvals1[256];histvals2[256];histvals[256];
FILE *stream;
time_t ltime,xx;
float dely,delx,f,af,bf,gf,hf,yi,delta;
struct videoconfig config;

/*Main program*/
main()
{

/*Set up the framegrabber board*/
     sethdw(0x100,0xD0000L,DUAL);
     setdim(512,512,8);
     fgon();
     initialize();
     setvmask(0x00);
     sclear(255);

/*Initial rectangle and LUT parameters*/
     a=30;b=30;da=450;db=450;grp=GREEN;
     loop:

/*Present menu on screen*/
     _clearscreen(_GCLEARSCREEN);
```

```
      printf("type:\n");
      printf("A for select mem A\n");
      printf("B for select mem B\n");
      printf("C for display mem A\n");
      printf("D for display mem B\n");
      printf("E for grab\n");
      printf("F for snap\n");
      printf("G for sclear(0)\n");
      printf("H for sclear(255)\n");
      printf("I to store image\n");
      printf("J to read image\n");
      printf("K to change setup\n");
      printf("L analysis\n");
      printf("M for store lut\n");
      printf("N for read lut\n");
      printf("O for select output lut\n");
      printf("P for select input lut\n");
      printf("Q for zoom\n");
      printf("R for unzoom\n");
      printf("S for sharpen\n");
      printf("T for low pass\n");
      printf("U for edge detector\n");
      printf("V for image average\n");
      printf("W for framegrabber off and exit program\n");

/*Get menu choice*/
      i=getch();
      i=tolower(i);
      _clearscreen(_GCLEARSCREEN);

/*Select memory option*/
      if(i=='a'){
      select_mem(MEM_A);
      goto loop;}

/*Select memory option*/
      else if(i=='b'){
      select_mem(MEM_B);
      goto loop;}

/*Display memory option*/
      else if(i=='c'){
      display_mem(MEM_A);
      goto loop;}

/*Display memory option*/
      else if(i=='d'){
      display_mem(MEM_B);
      goto loop;}

/*Continuous acquisition and display option*/
      else if(i=='e'){
      grab(-1);
      goto loop;}

/*Acquire and display one frame option*/
```

```
     else if(i=='f'){
     snap(WAIT);
     goto loop;}

/*Clear screen to black option*/
     else if(i=='g'){
     sclear(0);
     goto loop;}

/*Clear screen to white option*/
     else if(i=='h'){
     sclear(255);
     goto loop;}

/*Save image to disk option*/
     else if(i=='i'){
     printf("\n\n filename?\n");
     scanf("%s",name);
     printf("comment?\n");
     scanf("%s",comment);
     if((err==saveim(0,0,511,511,EIGHT_BIT,name,comment))<0){
        switch(err){
           case FILE_ERROR:
             printf("error opening file\n");
             break;
           case FORMAT_ERROR:
             printf("unknown format selected\n");
             break;
           default:
             printf("unknown error%d\n");
             break;}
     exit(1);}
     goto loop;}

/*Read image from disk option*/
     else if(i=='j'){
     printf("\n\n filename?\n");
     scanf("%s",name);
     if((err==readim(0,0,511,511,name,comment))<0){
        switch(err){
           case FILE_ERROR:
             printf("error opening file\n");
             break;
           case FORMAT_ERROR:
             printf("unknown file format\n");
             break;
           default:
             printf("unknown error\n");
             break;}
        exit(1);}
     printf("%s\n",comment);
     goto loop;}

/*Change memory and register base addresses in software option*/

     else if(i=='k'){
```

```
      regbase=0x100;membase=0xD0000L;
      printf("\n\n Default register base address is 0x100.\n");
      printf("Do you want to change it? type 'y' or 'n'\n");
      nnn=getch();
      if(nnn=='n')goto next1;
      if(nnn=='y'){
      printf("Enter new register base address. e.g. 100 \n");
      scanf("%x",&regbase);}
      next1:
      printf("\n\n Default memory base address is 0xD0000L.\n");
      printf("Do you want to change it? type 'y' or 'n'\n");
      res=getch();
      if(res=='n')goto next2;
      if(res=='y'){
      printf("Enter new memory base address.  e.g. D0000\n");
      scanf("%lx",&membase);}
      next2:
      sethdw(regbase,membase,DUAL);
      goto loop;}

/*LUT analysis option */
      else if(i=='l'){
      _setvideomode(16);
      select_mem(MEM_A);display_mem(MEM_A);grab(-1);
      loop160:
      _clearscreen(_GCLEARSCREEN);

      /*Display current LUT*/
      lutcoord();
      ralut(grp,0,0,256,array);
      displaylut();

      /*Present LUT analysis options*/
      _settextposition(1,1);
      printf("m=mod lut; n=mod2; h=histogram; e=equalize; l=linlut;
                i=invlut  ");
      _settextposition(2,1);
      printf("a=abslut; p=loglut; g=grab; f=freeze; s=stop;          ");

      /*Get LUT analysis choice*/
      c=getch();
      c=tolower(c);

      /*Continuous acquisition and display option*/
      if(c=='g'){select_mem(MEM_A);
      display_mem(MEM_A);grab(-1);goto loop160;}

      /*Acquire and display a single frame option*/
      else if(c=='f'){select_mem(MEM_A);
      display_mem(MEM_A);snap(WAIT);goto loop160;}

      /*Exit analysis option*/
      else if(c=='s')goto stop30;

      /*Histogram option*/
      else if(c=='h')goto part2;
```

```
/*Histogram equalization option*/
else if(c=='e'){histequal();goto loop160;}

/*Load linear LUT option*/
else if(c=='l'){linlut(grp,0);goto loop160;}

/*Load inverse LUT option*/
else if(c=='i'){invlut(grp,0);goto loop160;}

/*Load absolute value LUT option*/
else if(c=='a'){abslut(grp,0);goto loop160;}

/*Load logrithmic LUT option*/
else if(c=='p'){loglut(grp,0);goto loop160;}

/*Modify LUT using slope change option*/
else if(c=='n')goto mod;

/*Modify LUT by drawing option*/
else if(c=='m')goto part1;

else{goto loop160;}

/*LUT drawing routine*/
part1:
pp=180;qq=170;blx=pp;elx=blx-1;
loop150:
_settextposition(1,1);
printf("b=beginline; e=endline; d=drawline; p=pointer; s=stop;  ");
_settextposition(2,1);
printf("                                                        ");
_moveto(pp-2,qq);_lineto(pp+2,qq);
_moveto(pp,qq-2);_lineto(pp,qq+2);
c=getch();
c=tolower(c);
if(c=='3'){pp=pp+10;if(pp>304)pp=304;}
else if(c=='7'){pp=pp-10;if(pp<51)pp=51;}
else if(c=='1'){qq=qq+10;if(qq>299)qq=299;}
else if(c=='9'){qq=qq-10;if(qq<46)qq=46;}
else if(c=='6'){pp=pp+1;if(pp>304)pp=304;}
else if(c=='4'){pp=pp-1;if(pp<51)pp=51;}
else if(c=='2'){qq=qq+1;if(qq>299)qq=299;}
else if(c=='8'){qq=qq-1;if(qq<46)qq=46;}
else if(c=='b'){blx=pp;bly=qq;}
else if(c=='e'){elx=pp;ely=qq;}
else if(c=='s')goto loop160;
else if(c=='d')goto draw;
else if(c=='p'){pointer();grab(-1);}
else{goto loop150;}
goto loop150;
draw:
if(blx==elx)elx=blx+1;
_moveto(blx,bly);_lineto(elx,ely);
for(l=blx;l<=elx;l=l+1){
   dely=(float)bly-(float)ely;delx=(float)elx-(float)blx;
```

```
   f=(300-(float)bly)+((dely/delx)*(1-(float)blx));
   array[l-50 =(int)f;}
walut(grp,0,0,256,array);displaylut();
goto part1;

/*Slope changing routine*/
mod:
_settextposition(1,1);
printf("s=stop                                                    ");
_settextposition(2,1);
printf("                                                          ");
l=0;
mod1:
if(array[l+1]>0){a=l;goto mod2;}l=l+1;goto mod1;
mod2:
l=255;
mod3:
if(array[l-1]<255){b=l;goto mod4;}l=l-1;goto mod3;
mod4:
h=a;g=b;for(l=0;l<=255;l=l+1)array2[l]=array[l];
mod5:
c=getch();
c=tolower(c);
if(c=='7')g=g-1;
else if(c=='9')g=g+1;
else if(c=='1')h=h-1;
else if(c=='3')h=h+1;
else if(c=='s')goto loop160;
else{goto loop160;}
hf=(float)h;af=(float)a;bf=(float)b;gf=(float)g;
yi=(255*(hf-af))/(bf-af-gf+hf);
for(y=0;y<h;y=y+1)array[y]=0;
for(y=h;y<(int)yi;y=y+1){
   delta=((hf-af)/(yi-hf))*(yi-(float)y);
   array[y]=array2[(int)((float)y-delta)];}
for(y=(int)yi;y<g;y=y+1){
   delta=((gf-bf)/(yi-gf))*(yi-(float)y);
   array[y]=array2[(int)((float)y-delta)];}
for(y=g;y<=255;y=y+1)array[y]=255;
displaylut();walut(grp,0,0,256,array);goto mod5;

/*Histogram routine*/
part2:
_settextposition(1,1);
printf("d=default; r=rectangle;                                 ");
_settextposition(2,1);
printf("                                                        ");
c=getch();
c=tolower(c);
if(c=='d'){a=40;b=45;da=410;db=405;}
else if(c=='r'){mrectangle();grab(-1);}
else{goto part2;}
linlut(INPUT,0);linlut(grp,0);
select_mem(MEM_A);display_mem(MEM_A);snap(-1);
walut(INPUT,0,0,256,array);
select_mem(MEM_B);display_mem(MEM_B);snap(-1);
```

```
      linlut(INPUT,0);
      next45:
      _settextposition(1,1);
      printf("l=com linhist; m=com modhist;
               d=dis linhist; e=dis modhist;");
      _settextposition(2,1);
      printf("s=stop                                          ");
      c=getch();
      c=tolower(c);
      if(c=='l')comvals1();
      else if(c=='m')comvals2();
      else if(c=='d'){_clearscreen(_GCLEARSCREEN);lutcoord();displaylut();
         histcoord();plotvals1();linlut(GREEN,0);display_mem(MEM_A);}
      else if(c=='e'){_clearscreen(_GCLEARSCREEN);lutcoord();
         displaylut();histcoord();plotvals2();walut(GREEN,0,0,256,array);
         display_mem(MEM_A);}
      else if(c=='s'){linlut(GREEN,0);walut(grp,0,0,256,array);
          _settextposition(2,1);printf("       ");
          select_mem(MEM_A);display_mem(MEM_A);grab(-1);
          goto loop160;}
      else{goto next45;}
      goto next45;

      /*Exit analysis*/
      stop30:
      _settextposition(1,1);
      printf("                                                ");
      _settextposition(2,1);
      printf("                                                ");
      system("mode CO80");
      a=30;b=30;da=450;db=450;
      goto loop;}


/*Save LUT to disk option*/
      else if(i=='m'){
      printf("enter LUT file name\n");
      scanf("%s",lutname);
      save_lut(lutname,grp,0,0,256,NEW_FILE);
      goto loop;}

/*Read LUT from disk option*/
      else if(i=='n'){
      printf("enter LUT file name\n");
      scanf("%s",lutname2);
      read_lut(lutname2);
      goto loop;}

/*Select output LUT for loading option*/
      else if(i=='o'){
      grp=GREEN;
      goto loop;}

/*Select input LUT for loading option*/
      else if (i=='p'){
      grp=INPUT;
```

```
     goto loop;}

/*Zoom option*/
     else if(i=='q'){
     select_mem(MEM_A);display_mem(MEM_A);
     printf("\n\n To move upper left corner of rectangle type:\n\n");
     printf("'8' for up\n");
     printf("'2' for down\n");
     printf("'4' for left\n");
     printf("'6' for right\n\n\n");
     printf("type 'z' to zoom\n\n");
     loop70:
     while(!kbhit()){snap(WAIT);rectangle(a,b,240,235,0);}
     e=getch();
     e=tolower(e);
     if(e=='8'){b=b-10;if(b<0)b=479;}
     else if(e=='2'){b=b+10;if(b>479)b=0;}
     else if(e=='4'){a=a-10;if(a<0)a=512;}
     else if(e=='6'){a=a+10;if(a>512)a=0;}
     else if(e=='z')goto stop3;
     else{goto loop70;}
     goto loop70;
     stop3:
     pan(a);scroll(b);zoom(MAG_2);
     spot20:
     printf("\n\n type 'm' to return to menu\n");
     c=getch();
     c=tolower(c);
     if(c=='m'){da=240;db=235;goto loop;}
     else{goto spot20;}  }

/*Unzoom option*/
    else if(i=='r'){
    zoom(MAG_1);scroll(-b);pan(-a);
    select_mem(MEM_B);display_mem(MEM_B);sclear(255);
    select_mem(MEM_A);display_mem(MEM_A);sclear(255);
    initregs();setlut(grp,0);
     select_mem(MEM_A);display_mem(MEM_A);
     grab(-1);
     a=40;b=45;da=410;db=405;
     goto loop;}

/*Sharpen option*/

     else if(i=='s'){
     printf("\n\n\n\n\n\n\n\n\n\n\n
     PROCESSING IMAGE\n");
     sharpen(a,b,da,db,POSITIVE);
     goto loop;}


/*Low pass filter option*/
     else if(i=='t'){
     printf("\n\n\n\n\n\n\n\n\n\n\n
     PROCESSING IMAGE\n");
     lopass(a,b,da,db,POSITIVE);
```

```
     goto loop;}


/*Sobel edge filter option*/
     else if(i=='u'){
     printf("\n\n\n\n\n\n\n\n\n\n\n
     PROCESSING IMAGE\n");
     sobel(a,b,da,db,16);
     goto loop;}

/*Image average option*/
     else if(i=='v'){
     printf("number of frames?\n");
     scanf("%d",&num);
     _clearscreen(_GCLEARSCREEN);
     printf("\n\n\n\n\n\n\n\n\n\n\n
     PROCESSING IMAGE\n");
     average(10,10,490,470,num,num);
     goto loop;}

/*Framegrabber off option*/
     else if(i=='w')
     fgoff();

}

/*This function places a moveable pointer on video screen and feeds
position and intensity level to computer monitor*/
pointer(){
     select_mem(MEM_A);display_mem(MEM_A);
     _settextposition(1,1);
     printf("s=stop                                                    ");
     _settextposition(2,1);
     printf("                                                          ");
     pp1=255;qq1=240;
     loop50:
     snap(WAIT);
     intens=rpixel(pp1,qq1);
     _settextposition(2,65);
     printf("x=%d\n",pp1);
     _settextposition(3,65);
     printf("y=%d\n",qq1);
     _settextposition(4,65);
     printf("intensity=%d\n",intens);
     if(intens<=127)color=255;
     if(intens>127)color=0;
     x1=pp1-4;x2=pp1+4;yy1=qq1;y2=qq1;x3=pp1;x4=pp1;y3=qq1-4;y4=qq1+4;
     clr_line(x1,yy1,x2,y2,color);clr_line(x3,y3,x4,y4,color);
     while(!kbhit()){snap(WAIT);clr_line(x1,yy1,x2,y2,color);
     clr_line(x3,y3,x4,y4,color);}
     c=getch();
     c=tolower(c);
     if(c=='3'){pp1=pp1+10;if(pp1>511)pp1=0;}
     else if(c=='7'){pp1=pp1-10;if(pp1<0)pp1=511;}
     else if(c=='1'){qq1=qq1+10;if(qq1>479)qq1=0;}
     else if(c=='9'){qq1=qq1-10;if(qq1<0)qq1=479;}
```

```
     else if(c=='6'){pp1=pp1+1;if(pp1>511)pp1=0;}
     else if(c=='4'){pp1=pp1-1;if(pp1<0)pp1=511;}
     else if(c=='2'){qq1=qq1+1;if(qq1>479)qq1=0;}
     else if(c=='8'){qq1=qq1-1;if(qq1<0)qq1=479;}
     else if(c=='5')goto loop50;
     else if(c=='s')goto stop;
     else{goto loop50;}
     goto loop50;
     stop:;
     _settextposition(2,65);
     printf("              ");
     _settextposition(3,65);
     printf("              ");
     _settextposition(4,65);
     printf("              ");}

/*This function places a rectangle whose size and location may be
changed on the video screen.  Size and location parameters are then
used in other functions*/
mrectangle(){
     select_mem(MEM_A);display_mem(MEM_A);
     _settextposition(1,1);
     printf("s=stop                                                    ");
     _settextposition(2,1);
     printf("                                                          ");
     loop60:
     while(!kbhit()){snap(WAIT);rectangle(a,b,da,db,0);}
     e=getch();
     e=tolower(e);
     if(e=='8'){b=b-10;if(b<0)b=479;}
     else if(e=='2'){b=b+10;if(b>479)b=0;}
     else if(e=='4'){a=a-10;if(a<0)a=512;}
     else if(e=='6'){a=a+10;if(a>512)a=0;}
     else if(e=='9'){da=da+10;if(da+a>512)da=512-a;}
     else if(e=='7'){da=da-10;if(da<10)da=10;}
     else if(e=='3'){db=db+10;if(db+b>479)db=479-b;}
     else if(e=='1'){db=db-10;if(db<10)db=10;}
     else if(e=='s')goto stop2;
     else{goto loop60;}
     goto loop60;
     stop2:;}

/*This function performs histogram equalization and allows for manual or
automatic updates*/

histequal(){
     loop3:
     _settextposition(1,1);
     printf("d=default; r=rectangle;                                   ");
     _settextposition(2,1);
     printf("                                                          ");
     c=getch();
     c=tolower(c);
     if(c=='d'){a=40;b=45;da=410;db=405;}
     else if(c=='r'){mrectangle();grab(-1);}
     else{goto loop3;}
```

```
loop11:
_settextposition(1,1);
printf("c=continuous update; m=manual update;                    ");
_settextposition(2,1);
printf("                                                         ");
c=getch();
c=tolower(c);
if(c=='c')goto loop12;
else if(c=='m');
else{goto loop11;}
color=4;
loop2:
select_mem(MEM_B); display_mem(MEM_B);
snap(WAIT);
select_mem(MEM_A);
display_mem(MEM_A);
grab(-1);
select_mem(MEM_B);
histeq(grp,0,a,b,da,db);
ralut(grp,0,0,256,array);
_setcolor(color % 16);
displaylut();
color=color+5;if(color>9)color=4;
loop4:
 _settextposition(1,1);
printf("u=update; c=change param; f=freezeframe; g=grab; s=stop;");
_settextposition(2,1);
printf("                                                         ");
c=getch();
c=tolower(c);
if(c=='u')goto loop2;
else if(c=='c'){_setcolor(9 % 16);goto loop3;}
else if(c=='f'){snap(WAIT);goto loop4;}
else if(c=='g'){grab(-1);goto loop4;}
else if(c=='s'){_setcolor(9 % 16);goto stop25;}
else{goto loop4;}
loop12:
color=4;
_settextposition(1,1);
printf("enter time between updates (integer 2-9)                 ");
_settextposition(2,1);
printf("                                                         ");
c=getch();
if(c=='2')vt=2;else if(c=='3')vt=3;else if(c=='4')vt=4;
else if(c=='5')vt=5;else if(c=='6')vt=6;
else if(c=='7')vt=7;else if(c=='8')vt=8;else if(c=='9')vt=9;
else{goto loop12;}
_settextposition(1,1);
printf("c=change param; s=stop;                                  ");

_settextposition(2,1);
printf("                                                         ");
while(!kbhit()){
   time(&ltime);
   xx=ltime;
   select_mem(MEM_B);display_mem(MEM_B);
```

```
        snap(WAIT);
        select_mem(MEM_A);
        display_mem(MEM_A);
        grab(-1);
        select_mem(MEM_B);
        histeq(grp,0,a,b,da,db);
        ralut(grp,0,0,256,array);
        _setcolor(color % 16);
        displaylut();
        color=color+5;if(color>9)color=4;
        while((ltime-xx)<vt)time(&ltime);}
     c=getch();
     c=tolower(c);
     if(c=='c'){_setcolor(9 % 16);goto loop3;}
     else if(c=='s'){_setcolor(9 % 16);goto stop25;}
     else{_setcolor(9 % 16);goto loop3;}
     stop25:
     select_mem(MEM_A);
     display_mem(MEM_A);
     }

/*This function displays the current LUT shape on the computer monitor*/
displaylut(){
     c=_getcolor();
     _setcolor(2 % 16);
     _moveto(50,300);_lineto(305,45);
     _setcolor(c % 16);
     _moveto(50,300);
     for(l=50;l<=304;l=l+1){
        _lineto(l+1,300-array[l-49]);} }

/*This function draws the coordinates for the histogram display*/
histcoord(){
     _moveto(370,300);_lineto(370,40);_moveto(370,300);_lineto(630,300);
     _settextposition(23,47);printf("%d",0);
     _settextposition(23,58);printf("%d",100);
     _settextposition(23,71);printf("%d",200);
     _settextposition(15,43);printf("%d",2);
     _settextposition(8,43);printf("%d",4);
     for(l=370;l<=620;l=l+50){
        _moveto(1,300);_lineto(1,305);
        _moveto(370,670-1);_lineto(365,670-1);} }

/*This function computes a histogram for the contents of memory A*/
comvals1(){
     _settextposition(1,1);
     printf("                                                    ");
     _settextposition(2,1);printf("        ");
     _settextposition(13,50);printf("COMPUTING HISTOGRAM");
     select_mem(MEM_A);
     histogram(a,b,da,db,1,1,0,histvals1);
     _settextposition(13,50);printf("                    ");}

/*This function computes a histogram for the contents of memory B*/
comvals2(){
     _settextposition(1,1);
```

```
     printf("                                                    ");
     _settextposition(2,1);printf("        ");
     _settextposition(13,50);printf("COMPUTING HISTOGRAM");
     select_mem(MEM_B);
     histogram(a,b,da,db,1,1,0,histvals2);
     _settextposition(13,50);printf("                    ");}

/*This function plots the comvals1 output histogram*/
plotvals1(){
     _moveto(370,300);
     for(l=370;l<=625;l=l+1){
        _moveto(1,300);
        _lineto(1,300-(histvals1[l-370]/20));}}

/*This function plots the comvals2 output histogram*/
plotvals2(){
     _moveto(370,300);
     for(l=370;l<=625;l=l+1){
        _moveto(1,300);
        _lineto(1,300-(histvals2[l-370]/20));}}

/*This function draws the coordinates for the LUT display*/
lutcoord(){
     _moveto(50,300);_lineto(50,40);_moveto(50,300);_lineto(310,300);
     _settextposition(23,7);printf("%d",0);
     _settextposition(23,18);printf("%d",100);
     _settextposition(23,31);printf("%d",200);
     _settextposition(15,3);printf("%d",100);
     _settextposition(8,3);printf("%d",200);
     for(l=50;l<=300;l=l+50){
        _moveto(l,300);_lineto(l,305);
        _moveto(50,350-l);_lineto(45,350-l);}}
```

LIST OF REFERENCES

1. Ventura, Roberto M., *Analysis of an Image Processing Algorithm for its Implementation in Real Time*, Master's Thesis, Naval Postgraduate School, March 1988.

2. Imaging Technology Incorporated, *PCVISIONplus FRAME GRABBER USER'S MANUAL*, 1987.

3. Imaging Technology Incorporated, *ITEX PCplus PROGRAMMER'S MANUAL*, 1987.

4. Microsoft Corporation, *Microsoft C Run-Time Library Reference*, 1984-1987.

5. Pratt, William K., *Digital Image Processing*, John Wiley & Sons, 1978.

6. Gonzalez, Rafael C. and Wintz, Paul, *Digital Image Processing*, 2d ed., Addison-Wesley Publishing Company, 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Commander Naval Space Command Attn: Code N155 Dahlgren, Virginia 22448	1
4.	United States Space Command Attn: Technical Library Peterson AFB, Colorado 80914	1
5.	Director Navy Space Systems Division (OP-943) Washington, DC 20350-2000	1
6.	Superintendent Space Systems Academic Group, Code 72 Naval Postgraduate School Monterey, California 93943-5000	1
7.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
8.	Professor C. W. Therrien, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	3
9.	Professor Roberto Cristi, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	3
10.	Professor M. Fargues, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1

11. Professor Ralph Hippenstiel, Code 62 — 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California 93943

12. Professor Murali Tummala, Code 62 — 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California 93943

13. Mr. Richard Evans — 2
NUWES, Code 7021
Naval Undersea Weapons Engineering Station
Keyport, Washington 98345

14. CDR Hillier — 1
NUWES, Code 80
Naval Undersea Weapons Engineering Station
Keyport, Washington 98345

15. Mr. Alan L. Lindstrom — 1
NUWES, Code 70
Naval Undersea Weapons Engineering Station
Keyport, Washington 98345

16. Professor O. B. Wilson — 1
Department of Physics
Naval Postgraduate School
Monterey, California 93943

17. MAJ William J. Partridge — 3
Student Detachment
U.S. Army Command and General Staff College
Fort Leavenworth, Kansas 66027

Thesis
P24
c.1

Partridge
Real time image enhancement during underwater recovery operations.